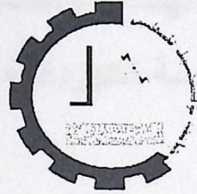


Palestine Polytechnic University



College of Engineering & Technology
Computer & Electrical Engineering Department

Graduation Project

Micro Web Based Energy Management System

Project Team

Omar Dhman Ibrahim Al-Sharif
Wajdy Zoughby

Project Supervisor

M. Elayan Abu-Gharbyeh

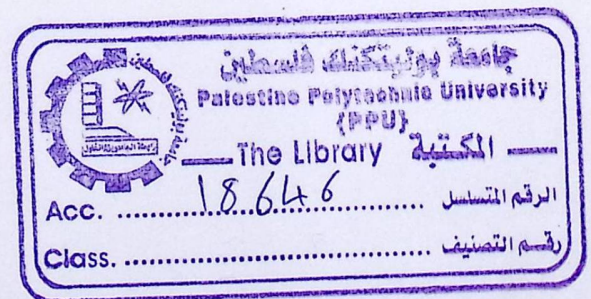
Project Co-Supervisor

ENG.Radwan Tahboub

Hebron - Palestine

June 2005

I




جامعة بوليتكنيك فلسطين
الخليل - فلسطين
كلية الهندسة والتكنولوجيا
دائرة الهندسة الكهربائية و الحاسوب

اسم المشروع:

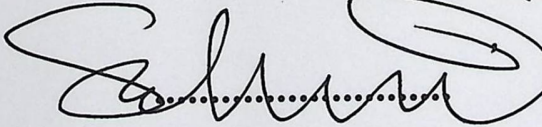
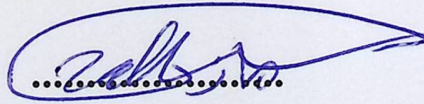
أسماء الطلبة
عمر دهمان إبراهيم الشريف
وجدي زغبى

بناء على نظام كلية الهندسة و التكنولوجيا و إشراف و متابعة المشرف المباشر على المشروع و موافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية و الحاسوب و ذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب

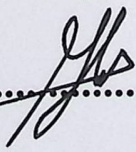
توقيع المشرف

م. عليان أبو غربية


توقيع اللجنة الممتحنة

توقيع رئيس الدائرة

د. غاندي المناصره


Dedication

To our Parents

To our Families

To our Friends

To our Supervisor

To all Lecturers in our University.....

To our University.....

Omar
Ibrahim
Wajdy

Acknowledgement

To our supervisor E.Elyan Abu-Garbyah for his guidance, support and encouragement, to every one who provides us with helpful suggestions.

Abstract

One of the technologies that is benefiting from the wide spread of the internet is the electrical energy management and monitoring. Controlling electrical energy remotely is improving very fast. This is in step with the wide spread of communication technologies, In our project we are going to build a small electrical management system, that has the capability of reading the active energy (KWH values) of two energy sources with ability to switch them either on or off. The system will provide these capabilities remotely using the internet. Our system makes use of a micro web server, the IPSIL 8930, to connect power control lines and counter units to the internet. The power control lines will be made using switch relays, and the counter units will be built using BCD counters; while the power meters will be an implementation of a pulse watt meters. We introduce this system as a step for making more advanced energy management projects.

تعد الطاقة الكهربائية من الأمور المهمة والأساسية في عالم اليوم، وأصبحت عملية إدارة الطاقة الكهربائية عن بعد وخصوصاً بعد الانتشار الواسع للإنترنت من الأمور ذات الأهمية الكبيرة والتي تساعد في متابعة والسيطرة على هذه الطاقة. في مشروعنا سنقوم ببناء نظام صغير لإدارة الطاقة، يحتوي هذا النظام على نظام للتحكم بالطاقة ونظام آخر لقراءة الـ Active Power لخطين جهد واحد فاز. يتكون نظام التحكم من Relays بينما يتكون نظام القراءة من نابض لقياس الطاقة مرتبط بعدادات لقياس الـ KWH المستخدمة من قبل الحمل، تتكون هذه العدادات من مجموعة BCDs والعداد الأخير مرتبط بـ Multiplexer للاختيار بين خطي الحمل. ونظام القراءة و التحكم متصلان بالإنترنت عن طريق خادم شبكة هو الـ IPSIL 8930. نقدم هذا المشروع كخطوة لمشاريع أكثر تقدماً في مجال إدارة الطاقة.

Table of Contents

<u>Title</u>	<u>Page</u>
Dedication	III
Acknowledgement	IV
Abstract	V
Table of Contents	VII
List of Figures	IX
List of Tables	XI
List of Abbreviations	XII

Chapter One

Introduction.....	2
Overview.....	2
1.2 Literature Review	3
1.2.1 PPU Projects for Micro web Server.....	3
1.2.2 PPU Projects for Power Meters.....	6
1.3 Estimated cost.....	8
1.4 Time plan.....	9
1.5 Report contents.....	9

Chapter Two

THEORETICAL BACKGROUND.....	12
2.1 TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL	12
2.2 java applet.....	14
2.3 IP μ 8930 Micro Web Server.....	16
2.3.1 General Description.....	16
2.3.2 IP μ 8930 Features.....	17
2.3.3 IP μ 8930 Developer Board Features.....	18
2.3.4 Micro-Web Server Control Protocol.....	19
2.3.5 IP μ 8930 Applications.....	19
2.4 Kilowatt-hour meter.....	20
2.4.1 Analog kiloWatt Hour Meter.....	20
2.4.2 Electronic Digital kiloWatt Hour Meter.....	21
2.4.3 How To Calculate Power	21

Table of Contents

<u>Title</u>	<u>Page</u>
Chapter Three	
Design Concepts.....	24
3.1 Project Objectives.....	24
3.2 Project General Block Diagram.....	24
3.3 How System works.....	26
Chapter Four	
Hardware System Design.....	28
4.1 AMR Automatic meter reading Options.....	28
4.2 The Micro Web Server.....	31
4.3 Metering units.....	34
4.3.1 Load circuit.....	35
4.3.2 Optocouplers circuit.....	35
4.3.3 Comparator circuit.....	36
4.3.4 Clock circuit.....	36
4.4 Counting unit.....	38
4.5 Controlling circuit.....	40
4.6 Backup battery unit.....	42
Chapter Five	
Software System Design.....	44
5.1 Software Parameters.....	44
5.2 Software Tools.....	45
5.3 System Flowcharts.....	46
5.3.1 System Authorization.....	46
5.3.2 Power Meter Viewing.....	47
5.3.3 Micro-Web Server Configuration.....	48
5.3.4 Read option.....	49
5.4 Web Pages Implementations.....	50
5.5 Implementing Java Applet procedures.....	51
5.6 System Interface.....	53

List of Figures

<u>Figure</u>	<u>Page</u>
Figure 2-1 TCP/IP compared to the OSI Model.....	13
Fig 2.2 Data movement through the TCP/IP layers.....	14
Fig 2.3: IP μ 8930.....	17
Fig 2.4: IP μ 8930 developer board.....	18
Fig 2.5 analog meter.....	20
Fig 2.6: Relatios between power.....	22
Fig 3.1 Project General Block Diagram.....	25
Fig 3.2 Pulse meter output and the Counter Unit.....	25
Fig 4.1: A kilo Watt hour meter with RS-485 communication.....	28
Fig 4.2: A kilo Watt hour meter with PC Parallel port interface.....	29
Fig 4.3: A Kilo Watt Hour meter with a pulse output.....	29
Fig 4.4: Energy Management System.....	30
Fig 4.5: IP μ 8930 Block Diagram.....	31
Fig 4.6: Parallel Connection.....	32
Fig 4.7: Serial connection.....	33
Fig 4.8: Block Diagram for meter circuit.....	34
Fig 4.9: Metering unit circuit.....	35
Fig 4.10: OptoCoplere circuit.....	36
Fig 4.11: Block Diagram For Clock Circuit.....	36
Fig 4.12: Clock circuit.....	37
Fig 4.13: Counter units with a multiplexer.....	38
Fig 4.14: Counter units with a multiplexer and Display.....	39
Fig 4.15: Block Diagram for Power Control Circuit.....	45
Fig 4.16: Power Control Circuit.....	41
Fig 4.13: Backup Battery Block Circuit.....	42
Fig 4.13: Backup Battery Circuit.....	42
Fig 5-1: Power Meter Viewing.....	46

Fig 5-2: Power Meter Viewing.....	47
Fig 5-3: Micro Web Server Configuration.....	48
Fig 5-4: Read Option.....	49
Fig 5-5: First Page.....	53
Fig 5-6: Verfyng User Account.....	54
Fig 5-7: Power Meter Value.....	55
Fig 6-1: BCD Counter Testing.....	70
Fig 6-2: Quad 2-1 Multiplexer Testing.....	70

List of Tables

<u>Table</u>		<u>Page</u>
Table 1.1: Hardware Cost.....		9
Table 1.2: Software Cost.....		10
Table 1.3: Schedule time for the system.....		10

List of Abbreviations

TCP/IP	Transmission Control Protocol/Internet Protocol
PPU	Palestine Polytechnic University
OSI	Open System Interconnection
HTTP	Hyper Text Transfer Protocol
IGMP	Internet Group Message Protocol
UDP	User Datagram Protocol
ICP	IP μ 8930 Control Protocol
LSI	Large-Scale Integration
kWh	Kilowatt Hour

Chapter One

Introduction

1.1 Overview

1.1 Overview

Remote monitoring and controlling over the internet is increasing and it became more valuable for applications to be connected to the internet and monitoring them remotely.

1.2 Literature Review

Many organizations are showing great concern for redesigning their services and making them more efficient. This will improve the organization services and will open the door for a wide variety of services.

1.3 Estimated cost

1.4 Time plan

One of the things that might have been raised up is the design of an energy management system. This system should have many services ranging from monitoring the electrical consumption and the energy parameters including power, current and voltage and controlling them remotely.

1.5 Report contents

Many designs were created, these will be discussed in chapter 2. Our project will be a design of a small management system providing the services for monitoring the active power that is consumed by a two load systems with the ability to control the supply voltage delivered to them.

The system should provide these services remotely so our design includes a web server that will be introduced to the power controlling and measuring circuits and then connected to the internet.

Chapter One

Introduction

1.1 Overview

Nowadays internet is spreading very fast it has entered a very huge number of places from small houses to government buildings. Remote monitoring and controlling over the internet is increasing too, and it became more valuable for applications to be connected to the internet and so controlling and monitoring them remotely.

Many organizations are showing great concern for redesigning their services and making them available online. This will improve the organization services and will open the door for a wide variety of services.

One of the things that might have been raised up is the design of an energy management system. This system should have many services ranging from monitoring the electrical components and the energy parameters including power, current and voltage and many others to controlling them remotely.

Many designs were created; these will be discussed in chapter 2. Our project will be a design of small management system providing the services for monitoring the active power that is consumed by a two load systems with the ability to control the supply voltages delivered to them.

The system should provide these services remotely so our design includes a web server that will be interfaced to the power controlling and measuring circuits and then connected to the internet.

1.2 Literature Review

This project is the fourth phase of the Micro Web Server projects. A TCP/IP stack was designed in the first phase. A home Automation System was implemented in the second phase. A Digital camera and two stepper motors controlled by the Micro Web Server in the third phase. A power meters will be interfaced and controlled by the Micro Web Server in this phase.

We can categorize the Literature Survey into two main groups:

1.2.1 PPU Projects for Micro web Server

As we had mentioned it's the fourth phase for the project idea, the three previous projects that tackled this issue are described here briefly:

[A] Project title: "Micro Web-Server"

Project Overview:

This project was introduced and approved in the 2001/2002 academic year in our university. The project idea was to build a Micro Web-Server from its primitive components and use it to perform simple controlling aspects such as LED's switching.

Project Description:

The aim of the project was to write the smallest possible TCP/IP stack, which can communicate with any web browser over the Internet. The web browser must translate the web server files and fetch them with its large screen, user-friendly menus and buttons and helpful information which guide the user through a process by just a mouse click.

The project was focusing on:

- Understanding the fundamentals of computer-internetworking and how the internet works.
- Understanding the Open System Interconnection (OSI) model which was developed by the International Organization for standardization, and learning the details of the TCP/IP, HTTP and more networking issues.
- Building the TCP/IP stack which includes implementations of the TCP, IP, ICMP and HTTP v1.1.
- Hardware designing and implementation.

Similarities with our project:

Although our proposed project will optimize a Micro Web Server to accomplish the hardware controlling and the software management, but it will eliminate a lot of the work done in that project because of the following:

- 1) We will use an off-the-shelf Micro Web Server rather than building it.
- 2) We will use Java programming language and java applets to program the system rather than writing C codes.
- 3) The application in our project is widely different and complicated.

[B] Project Title: "Micro-Web Server for controlling and monitoring applications"

Project Overview:

This project was introduced and approved in the 2002/2003 academic year in our university. It used the IPU8930 Micro-Web Server and the Java Applets software to control some simple home applications.

Project Description:

The project is considered as a home automation system, in which Micro Web Server controls and manages many 'dumb' (non-computerized) devices. It is a system which was designed to control various devices, these devices are:

- Lightening System.
- Temperature Control.
- Alarm System.
- Fan Controlling.

Similarities with our project:

We think that this project is strongly related to our project, this accentuation stems from the following facts:

- 1) We will exploit the same Micro Web Server.
- 2) Most often we will use java applets as a programming environment.

But there still general schemes that may use new ways, such as:

- 1) Application is different.
- 2) Ports usage.

[C] Project Title: "Controlling a Digital Camera Using Micro Web Server"

Project Overview:

This project was introduced and approved in the 2003/2004 academic year in our university. It used the IPu8930 Micro-Web Server and the Java Applets software to control the Digital by the serial port RS-232.

Project Description:

The aim of the project was to control the operation and position of the camera.

The project was focusing on:

- Understanding the fundamentals of the Digital camera and how the sequence of the operation process is done.
- Understanding the Open System Interconnection (OSI) model which was developed by the International Organization for standardization, and learning the details of the TCP/IP, HTTP and more networking issues.
- Hardware designing and implementation.

Similarities with our project:

We think that this project is the most related project to us, this comes from the fact that:

- 1) We will use the same Micro Web Server.
- 2) Most often we will use java applets as a programming environment.

But there still general schemes that may use new ways, such as:

- 1) Application is different.
- 2) They used serial port, but we will use parallel port.

1.2.2 PPU Projects for Power Meters

Our project is the first project concerning power meters management and interfacing

Other Projects:

Through searching the Internet, we had seen various and different projects and suggestions tackling the idea of our project, some of them are tightly related to it, such as:

1) For Micro web Server

"Micro Web server for domestic application"

It is a final year project in Newcastle University in Australia, the two goals of the project were:

- Miniaturization of computer; this includes microprocessor, microcontroller and other peripheral devices.
- The development of the Internet.

There is an AVR microcontroller and an Ethernet controller in that board. There is also an OS available for the board as well as its TCP/IP stack. The project controls devices like light-bulb and TV remotely

2) For Power Meter

"Electric Meter Reading via Bluetooth"

An architecture of a micro controller based system for interfacing between the Digital Energy Meter and the Bluetooth Module is proposed. An embedded instrumentation system onboard reads the Active Energy Register values from the Digital Energy Meter. This information are then returned to the host via a wireless Bluetooth link, which will be convert to an active energy value in KWH and Display in the host's GUI. An instantaneous energy consumption graph will be plot in the host GUI.

Two interesting applications have been identified. They are Automatic Meter Reading (AMR) and Automatic Polling Mechanism (APM). AMR is a mechanism whereby the Bluetooth Energy Meter sends the recorded energy consumption of a household in the certain interval of time to a 'wirelessly' connected reader. APM is a Feature where a reader will poll each and every individual Bluetooth Digital Energy Meter automatically in order to get the meter reading of the corresponding households.

"Power Meter Reading through internet connection"

Its objective was to read the power meter value through the internet, because they used a power meter that has a direct connection to the internet.

1.3 Estimated cost

In this project we need the following equipments in order to implement the system. The costs are divided into: hardware costs and software costs

Num.	H.W Requirements	The price
1	IPU8930 Micro Web-Server	\$200
2	Power Meter	\$100
Total		\$300

Table 1.1: Hardware Cost

Num.	S.W	The price
1	Software Implementation	\$200
2	Internet Domain	\$80
total		\$280

Table 1.2: Software Cost

1.4 Time plan

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1	3 Weeks															
T2			2 Weeks													
T3					4 Weeks											
T4								3 Week								
T5												3 Week			1 week	
T6	16 Weeks															

Table 1.3: Schedule time for the system

1.5 Report contents

This document consists of seven chapters describing the logical or physical part of the system

- **Chapter One: Introduction**

This chapter demonstrates an overview about the system, system design options, a literature review, estimated cost and time planning.

- **Chapter Two: Theoretical Background**

This chapter focuses on theories and materials that are related to our system operation and behavior. It mainly talks about TCP/IP protocol, Java Applets, Micro Web-Server and Power Metering System.

- **Chapter Three: Design Concepts**

This chapter describes the system in its abstract formula. It describes the project objectives, a general block diagram and how the system works.

- **Chapter Four: Hardware System Design**

This chapter discusses system design options and justifies those that are chosen in the project. A detailed description about the different project parts also is included.

- **Chapter Five: Software System Design**

This chapter handles the software related to our system, depicts flow charts about system operation and illustrates different algorithms and techniques that will be considered in writing the software.

- **Chapter Six: System Implementation and testing**

This chapter includes the implementation phases with the testing of these phases. General hardware and software components are tested and shown in this chapter.

- **Chapter Seven: Conclusion and Future Work**

This chapter will list the problems faced us in accomplishing the system and how did they resolved. Notes and Conclusions will help readers are also included. A future work is also proposed.

Chapter Two

Theoretical Background

2.1 TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL (TCP/IP)

2.2 java applet

2.3 IPμ8930 Micro Web Server

2.4 Kilowatt-hour meter

Chapter Two

Theoretical Background

2.1 TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL (TCP/IP)

TCP/IP is not simply one protocol, but rather a suite of small, specialized protocols including TCP, IP, UDP, ARP, ICMP, and others called subprotocols. Most network administrators refer to the entire group as "TCP/IP," or sometimes simply "IP." TCP/IP's roots lie with the U.S. Department of Defense, which developed the precursor to TCP/IP for its Advanced Research Projects Agency network (ARPAnet) in the late 1960s. Because of its low cost and its ability to communicate between a multitude of dissimilar platforms, TCP/IP has grown extremely popular. It is a de facto standard on the Internet and is fast becoming the protocol of choice on LANs.

The latest network operating systems (such as NetWare 5.x and Windows XP) use TCP/IP as their default protocol. The greatest advantages to using TCP/IP relates to:

- 1- Its status as a routable protocol which means that it carries network addressing information that can be interpreted by routers.
- 2- TCP/IP is also a flexible protocol, running on any combination of network operating systems or network media.
- 3- TCP/IP is a useful reference for understanding other protocols because it includes elements that are representative of other protocols.
- 4- Because of its flexibility, however, TCP/IP may require significant configurations.

The TCP/IP suite of protocols can be divided into four layers that roughly correspond to the seven layers of the OSI Model, as depicted in Figure 2-1 and described in the following list

OSI Model		TCP/IP Model
Application	-----	Application
Presentation		
Session		
Transport		Transport
Network		Internet
Data Link	-----	Network Interface
Physical		

Figure 2-1 TCP/IP compared to the OSI Model

Application layer—roughly equivalent to the Application, Presentation, and Session layers of the OSI Model. Applications gain access to the network through this layer, via protocols such as the File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and Dynamic Host Configuration Protocol (DHCP).

Transport layer—roughly corresponds to the Transport layer of the OSI Model. This layer holds the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), which provide flow control, error checking, and sequencing. All service requests use one of these protocols.

Internet layer—Equivalent to the Network layer of the OSI Model. This layer holds the Internet Protocol (IP), Internet Control Message Protocol (ICMP), Internet Group Message Protocol (IGMP), and Address Resolution Protocol (ARP). These protocols handle message routing and host address resolution.

Network Interface layer—roughly equivalent to the Data Link and Physical layers of the OSI Model. This layer handles the formatting of data and transmission to the network wire.

The movement of a packet of data through the layers in a TCP/IP network is shown in Figure 2.2. When a packet of data is sent, it travels to the transport layer where the transport header is added. Next the internet layer adds its header. Finally, the physical layer attaches its header. When a packet of data is received, the process is reversed, resulting in the application's reception of the intended data.

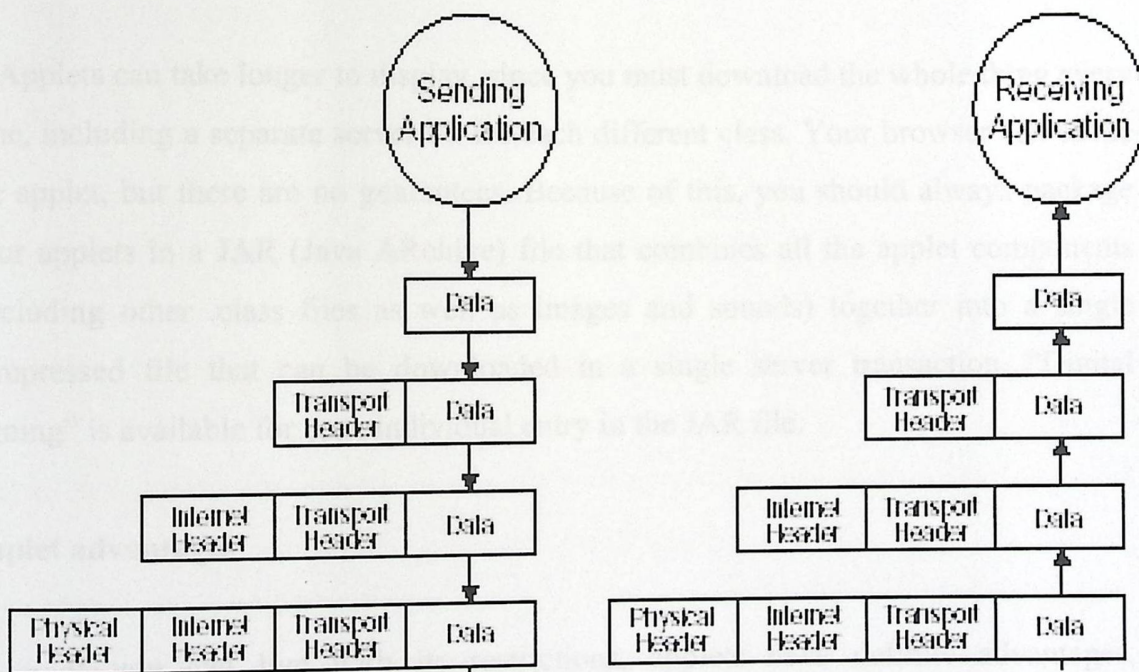


Fig 2.2: Data movement through the TCP/IP layers.

2.2 java applet

Java has the ability to create applets, which are little programs that run inside a Web browser. Because they must be safe, applets are limited in what they can accomplish. However, applets are powerful tools that support client-side programming, a major issue for the Web.

Looking at what it is supposed to do: extend the functionality of a Web page in a browser. Since, as a Net surfer, you never really know if a Web page is from a friendly place or not, you want any code that it runs to be safe. So the biggest restrictions you'll notice are probably:

1. An applet can't touch the local disk. This means writing or reading, since you wouldn't want an applet to read and transmit private information over the Internet without your permission. Writing is prevented, of course, since that would be an open invitation to a virus. Java offers digital signing for applets. Many applet restrictions are relaxed when you choose to allow signed applets (those signed by a trusted source) to have access to your machine.

2. Applets can take longer to display, since you must download the whole thing every time, including a separate server hit for each different class. Your browser can cache the applet, but there are no guarantees. Because of this, you should always package your applets in a JAR (Java ARchive) file that combines all the applet components (including other .class files as well as images and sounds) together into a single compressed file that can be downloaded in a single server transaction. "Digital signing" is available for each individual entry in the JAR file.

Applet advantages

If you can live with its restrictions, applets have definite advantages, especially when building client/server or other networked applications:

1. There is no installation issue. An applet has true platform independence, so you don't need to make any changes in your code for different platforms nor does anyone have to perform any installation "tweaking." In fact, installation is automatic every time the user loads a Web page that contains applets, so updates happen silently and automatically. In traditional client/server systems, building and installing a new version of the client software is often a nightmare.

2. You don't have to worry about bad code causing damage to someone's system, because of the security built into the core Java language and applet structure. This,

along with the previous point, makes Java useful for so-called intranet client/server applications that live only within a company or restricted area of operation where the user environment (Web browser and add-ins) can be specified and/or controlled.

2.3 IP μ 8930 Micro Web Server

2.3.1 General Description

The IP μ 8930 is a general purpose network controller and web server which makes it easy to monitor, control and communicate with remote sensors, actuators, and practically any devices with a serial port (via the onboard serial port) via a TCP/IP network such as the Internet. The IP μ 8930 is designed to either work standalone or in conjunction with a "master" MCU.

At the heart of the IP μ 8930 Developer Kit is the IP μ 8930 Module, a compact TCP/IP network controller and webserver module, which enables developers to rapidly add network connectivity to products. The IP μ 8930 combines a TCP/IP controller, HTTPcompliant webserver, Modbus TCP node and A/D converter into a single, small (1.3x1.4"/3.3x3.4cm) daughterboard. The IP μ 8930 is designed to enable remote monitoring and control over a TCP/IP-based network without the overhead and complexity of traditional solutions which require the knowledge on how to support and program using a real-time operating systems. The IP μ 8930 Module is capable of operating standalone or as a TCP/IP peripheral for a separate MCU.

In addition to being able to store and retrieve web pages on the IP μ 8930, developers can access the I/O ports on the IP μ 8930 using Modbus TCP, a popular standard for accessing devices on a network in the industrial automation, building automation, and utilities industries.

The IP μ 8930 Module plugs easily into the IP μ 8930 Developer Board. The Developer Board is packaged with power, a 10BaseT network jack, and various connectors to make it easy to set up, program, and test the IP μ 8930

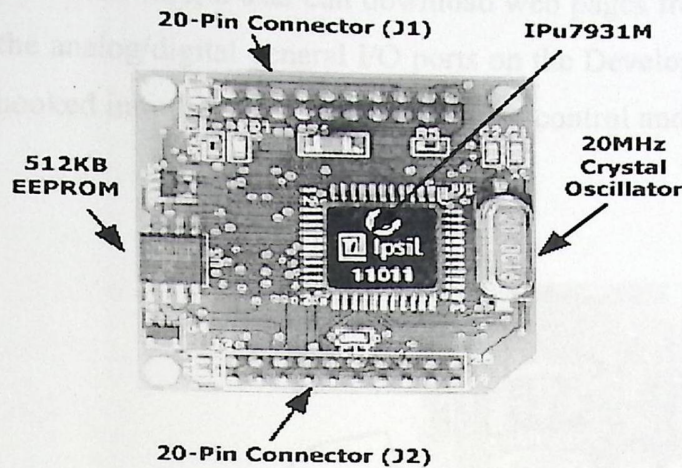


Fig 2.3: IP μ 8930

2.3.2 IP μ 8930 Features

The IP μ 8930 Board provides the following features:

- Supports both TCP/IP and UDP—read & write
- Includes HTTP v1.0 compliant embedded webserver
- Up to 8 analog or digital ports for monitoring/control of external devices
- Built-in 10-bit ADC
- Serial port
 - With flow control (CTS/RTS, XON/XOFF, or None)
 - Can be used to communicate with external serial devices
- Ability to define WebHoles™ in HTML pages which get “filled” with values from pre-defined I/O port
- Can operate standalone or as a peripheral to an external MCU using a serial interface
- Full set of user-configurable network and system settings including
 - Support for static & dynamic IP addresses
 - User-definable default fixed address
- Compliant with RFC-1122, the standard for TCP/IP hosts on the Internet
- Supports the following network applications:
 - ARP, DHCP, ICMP (ping), Modbus, and Ipsil Control Protocol (v1.0)
- 4 megabit (512KB) of on-board memory for web pages.
- Time stamp functionality
- File system with over 500 unique, variable length files

Without any additional hardware, a user can download web pages from the IP μ 8930 Board and control the analog/digital general I/O ports on the Developer Board. With additional devices hooked into the Serial ports, a user can control and monitor a wide variety of devices.

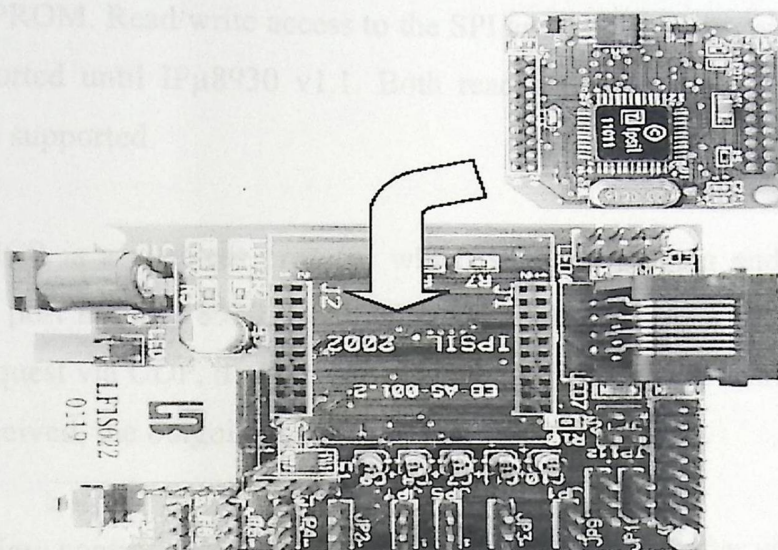


Fig 2.4: IP μ 8930 developer board

2.3.3 IP μ 8930 Developer Board Features

The IP μ 8930 Developer Board is designed to help you quickly develop applications using the IP μ 8930 module. The IP μ 8930 Developer Board contains the following features:

- Ethernet connector with two integrated LEDs which shows Rx/TX activity
- RS-232 connectivity thru a serial RJ-45 jack (RJ-45 to DB9 cable included) with flow control
- 8 general I/O channels accessible through developer board header
- Service LEDs 5V, 3.3V, Tx, Rx, RTS, CTS activity
- Integrated I2C temperature sensor
- Reset button
- Power and collision activity LED
- External power jack

2.3.4 Micro-Web Server Control Protocol

The IP μ 8930 Control Protocol (ICP) provides programmatic access to all of the I/O and storage resources on the IP μ 8930 Board including the general I-O pins (a.k.a., Port A), the Serial Port, IP μ 8930 RAM, the internal EEPROM, and the External EEPROM. Read/write access to the SPI and I2C ports is noted here but will not be supported until IP μ 8930 v1.1. Both read and write operations to IP μ 8930 resources are supported.

The ICP is a stateless protocol which supports reading and writing to the IP μ 8930 via port number 8930. Both TCP and UDP are supported. If the IP μ 8930 receives a request via UDP, it will reply to the request using UDP. Similarly, if a TCP request is received, the outgoing packet will be a TCP packet.

The flow control for ICP is lock-step. A program must, after writing a packet, Wait for a reply before sending another packet.

The ICP packet format consists of several parts. First, every write packet starts with a password, which is required if secure mode is enabled—otherwise it is Ignored. An OPCODE and address are next required. The remaining fields carry the information read from or to be written to the IP μ 8930. The contents of the data structure differ based on the operation and whether this is a request or response.

Table's details of the packet of an ICP read, an ICP write and there respective responses are described in appindex.

2.3.5 IP μ 8930 Applications

Applications for IP μ 8930 Modbus include:

- Remote monitoring
- Industrial automation and process control
- Home automation
- HVAC control

- Lighting control
- Environmental monitoring
- Remote telemetry
- Test and lab equipment monitoring

2.4 Kilowatt-hour meter

Kilowatt-hour meter is a device that is used to measure the number of kilowatt power used by load in a unit of time. kilowatt hour meter may be used for three phase or single phase reading, we here will talk about single phase meter only.

There are two types of kilowatt-hour meter using for measuring power:

- Analog KiloWatt Hour Meter.
- Electronic Digital KiloWatt Hour Meter.

2.4.1 Analog kiloWatt Hour Meter:

This kind of kilowatt hour meter has a mechanical counting system that counts the revolutions of an aluminum disk and in turn, displays the kilowatts of energy used. The ratio of kilowatts displayed to the turns of the disk is called the disk constant, and this constant is different from one meter to other depending on the rotating disk.

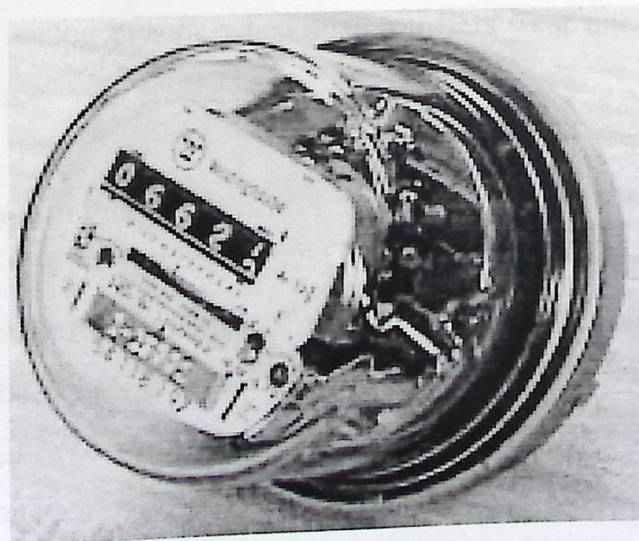


Fig 2.5 analog meter

2.4.2 Electronic Digital kiloWatt Hour Meter

The digital meters determine power by directly multiplying the current by the potential difference and record this result over time to obtain a value for energy used.

There are a few advantages of using a digital meter over an analog meter. The primary reason is the fact that digital meters do not have moving parts. This reduces the error due to mechanical parts wearing out after time. Also a digital meter is more accurate; most digital meters have an error of 0.8%. While the error of a standard, analog meter is at most 2%. Some digital meters are designed using large-scale integration circuits (LSI). LSIs contain anywhere from 500 to 1000 circuit elements. An LSI uses a digital voltmeter and an ammeter to measure the instantaneous potential difference and current. Then it is capable of converting the values into computer code. After that, it performs the necessary multiplication to obtain the wattage used and records the amount in static memory.

Another advance in the digital kilowatt-hour meter is the ability to transmit the current readings to a remote observer. With this capability, the power company can effectively monitor all the meters from a central location instead of physically looking at each meter individually.

Even though the newer, digital meters are more reliable they are still not widely used. Although the benefits of a digital meter don't warrant an immediate replacement of all analog meters, when analog meters wear out, digital versions are installed in their place.

2.4.3 How to Calculate Power

A measurement for a quantity of electric energy, at any point in time, is called a "watt". 1000 watts equals a "kilowatt" or simply kW. This energy measured over time is called a "Kilowatt-Hour" or "kWh". One kWh means the usage of 1 kW for one hour or sixty minutes.

Active energy is measured in kilowatt-hours, while reactive and apparent energy are VAR hours and VA hours, respectively. Figure 2.6 shows the relationship between active, reactive, and apparent energy. The relationship described in the figure holds true for pure sinusoids at the fundamental frequency. In the presence of harmonics, this relationship is not valid.

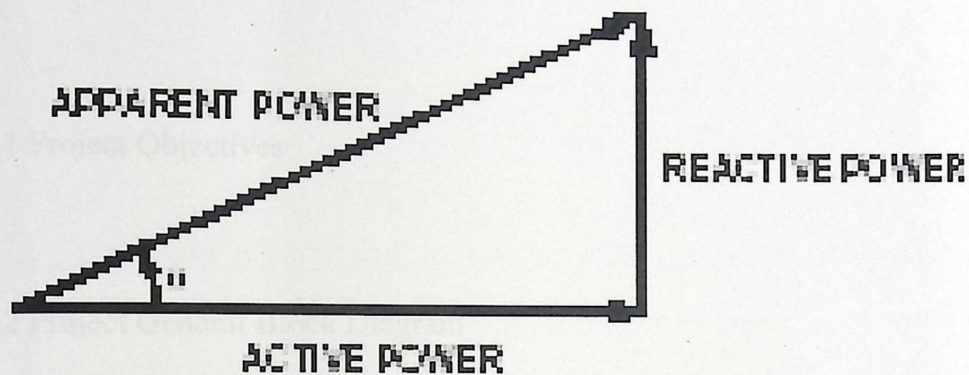


Fig 2.6: Relations between powers

The relationships are as follows:

- Active Power = $VI \cos \theta$
- Reactive Power = $VI \sin \theta$
- Apparent Power = VI
- Power Factor = $\cos \theta$

Chapter Three

Design Concepts

3.1 Project Objectives

The project to be designed should have following objectives:

1. To control multiple AC loads using the internet.
2. To be able to read KWH for the AC loads using the internet.

3.2 Project General Block Diagram

3.3 How System works

Figure 3.1 shows the system (to be designed) in a block diagram. This system consists of the following components:

- AC Lines
This project assumes two separate AC lines, one for small loads such as lighting, and the other for small loads, such as refrigerators, motors.
- Metering units
Sense the power consumed by the loads.
- Power control units
Switch off or on the ac loads.
- Computer units
Counts the consumed KWH (kilo-watt hour value).

Chapter Three

Design Concepts

3.1 Project Objectives

The project to be designed should have following objectives:

1. Design a Power Meter control and management system.
2. To control multiple AC lines using the internet.
3. Be able to read KWH for the AC lines using the internet.

3.2 Project General Block Diagram

Figure 3.1 shows the system (to be designed) in a block diagram; this system consists of the following components:

- AC Lines
This project assumes two separate AC lines, one for small loads such as lighting, and the other for small loads, such as refrigerators, motors.
- Metering units
Sense the power consumed by the loads.
- Power control units
Switch off or on the ac lines.
- Counter units
Counts the consumed KWH (kilo watt hour value).

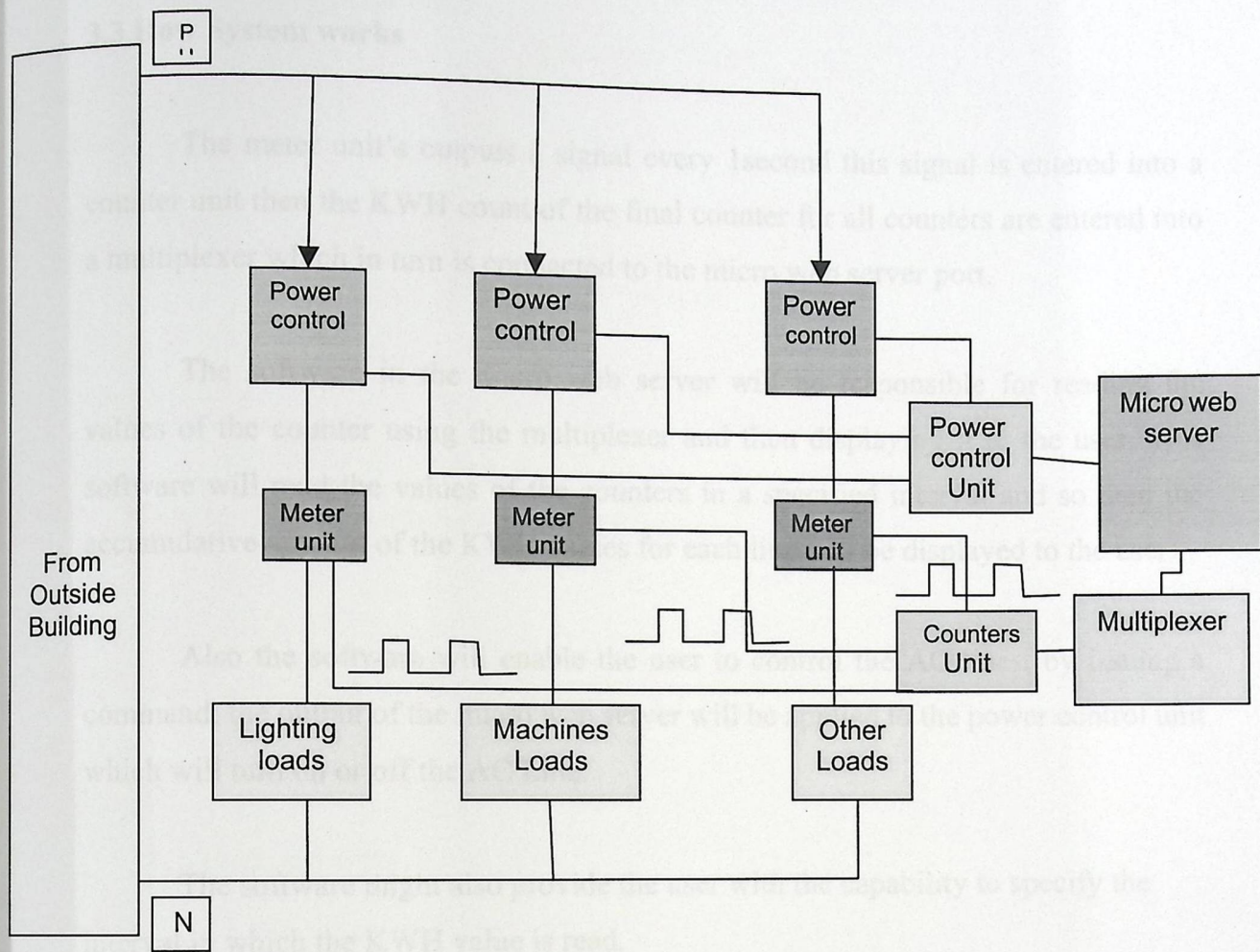


Fig 3.1 Project General Block Diagram

The general block diagram describes the system for three loads, but the system design will be for two loads

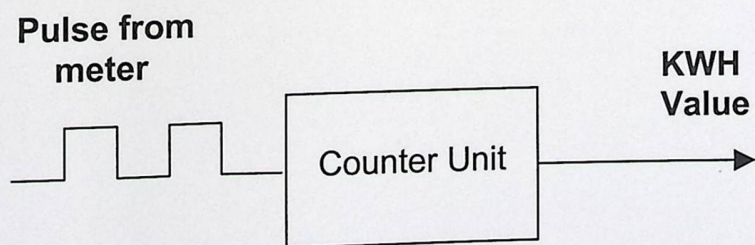


Fig 3.2 Pulse meter output and the Counter Unit

3.3 How System works

The meter unit's outputs a signal every 1second this signal is entered into a counter unit then the KWH count of the final counter for all counters are entered into a multiplexer which in turn is connected to the micro web server port.

The software in the micro web server will be responsible for reading the values of the counter using the multiplexer and then displaying it to the user. This software will read the values of the counters in a specified interval and so then the accumulative amount of the KWH values for each line will be displayed to the user.

Also the software will enable the user to control the AC lines, by issuing a command; the output of the micro web server will be applied to the power control unit which will turn on or off the AC Line.

The software might also provide the user with the capability to specify the interval in which the KWH value is read.

Chapter Four

Hardware System Design

In this chapter we will describe the hardware components of our design.

4.1 AMR Automatic meter reading Options

4.2 The Micro Web Server

Types of the interface supported by the meter

4.3 Metering units

- PC Parallel port interface
- Pulse output interface

4.4 Counting unit

- RS-485 interface

4.5 Controlling circuit

depends on the interface supported by the meter. A lot of

interfaces are available these includes the following

- RS-485 interface, as example is shown in the figure below

4.6 Backup battery unit.

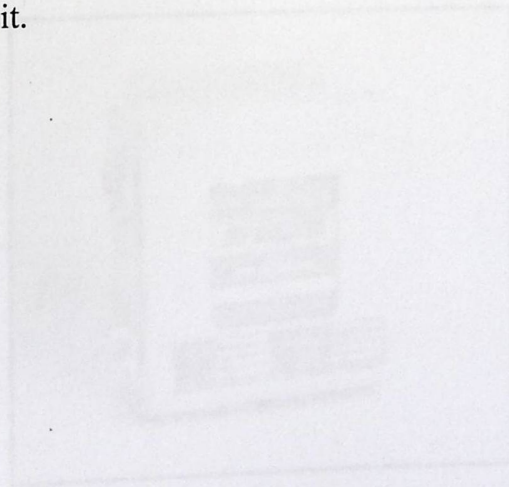


Fig 4.1. A 4x16 Watt hour meter with RS-485 communication

Chapter Four

Hardware System Design

In this chapter we will describe the hardware components in our design system, and how these components integrated to each other.

4.1 AMR Automatic meter reading Options

Types of the interface supported by the meter

- RS-485 Interface
- PC Parallel port interface
- Pulse output interface
- RS-232 interface
- RJ-11 interface

The reading of meters depends on the interface supported by the meter. A lot of interfaces are available these includes the following

- RS-485 Interface, an example is shown in the figure below.



Fig 4.1: A kilo Watt hour meter with RS-485 communication

- PC Parallel port interface, the figure below shows a functional block diagram of a KWH meter that is build on the analog device energy IC ADE7756

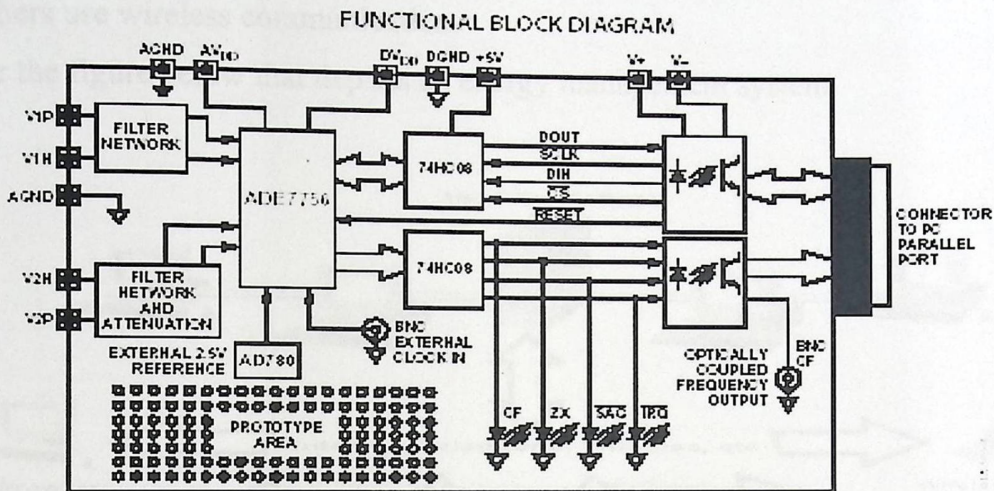


Fig 4.2: A kilo Watt hour meter with PC Parallel port interface

- Pulse output interface, see the figure below.

Models are available for single phase, three phase three wire and three phase four wire systems. Inputs are from standard current transformers and normal mains voltages. Output is in the form of unit energy (watt-hour) pulses read by the built in counter or outputted to an external counting system.

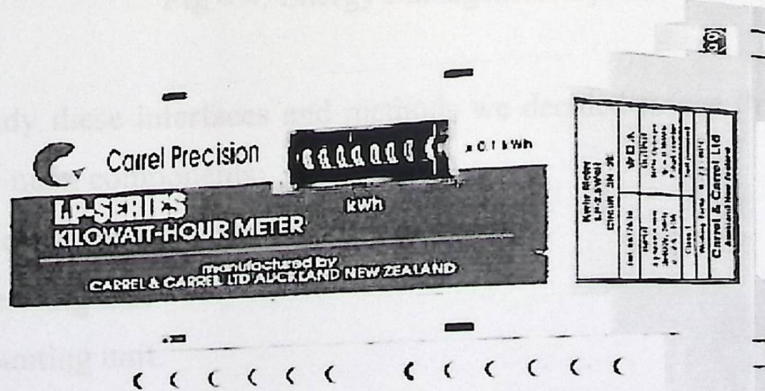


Fig 4.3: A Kilo Watt Hour meter with a pulse output

A lot of meters are also supported with software for the controlling and monitoring of the meters.

There are a lot of ways for gathering information related to metering from various locations; some of them take the power lines as a means for communications while others use wireless communication.

Consider the figure below that depicts an energy management system.

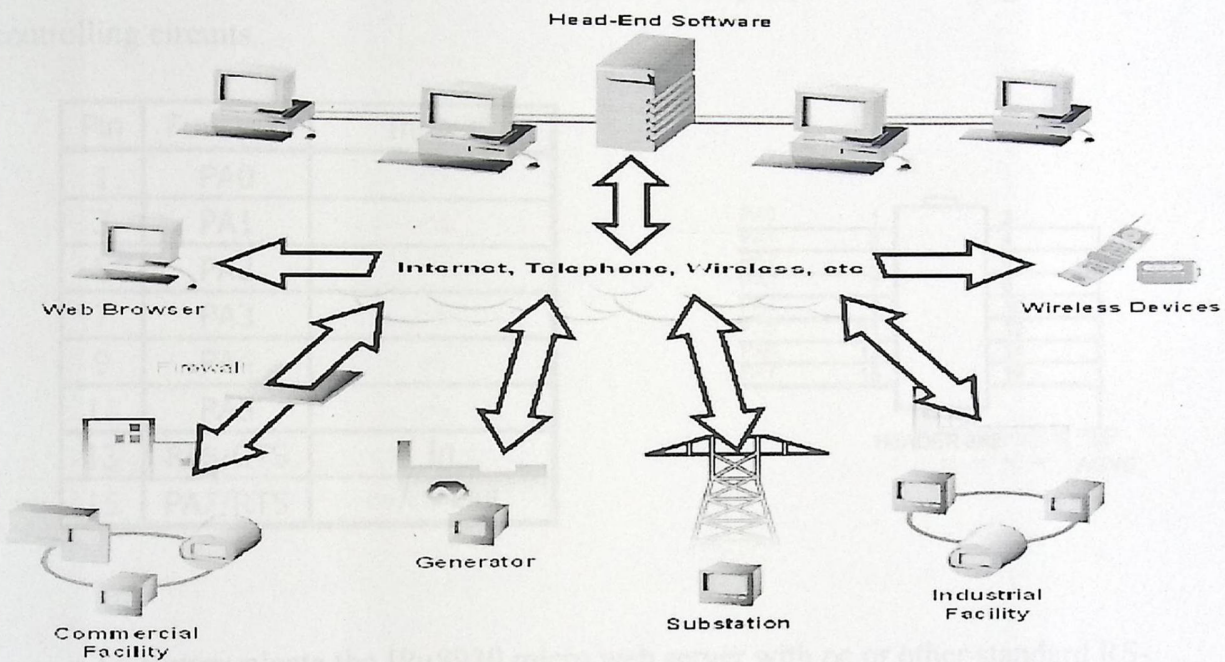


Fig 4.4: Energy Management System

After we study these interfaces and methods we decided to use this system which contains five main components:

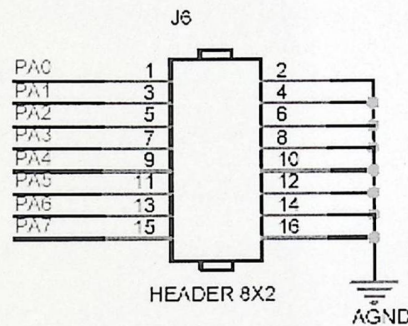
1. Micro Web Server
2. Metering unit
3. Counting unit.
4. Controlling unit.
5. Backup battery unit.

4.2 The Micro Web Server

General Purpose I/O Pins (Port A)

The IP μ 8930 micro web server contains 8 pins, each pin can be used for input (set it to 1) or output (set it to 0), each pin can also be configured as analog or digital. In this system we use these as digital I/O, four pins will be used as input bins to read from counting circuits, and other bins will be used for output pins for selecting and controlling circuits.

Pin	Function	In/Out
1	PA0	\leftrightarrow
3	PA1	\leftrightarrow
5	PA2	\leftrightarrow
7	PA3	\leftrightarrow
9	PA4	\leftrightarrow
11	PA5	\leftrightarrow
13	PA6/CTS	\leftrightarrow / In \leftarrow
15	PA7/RTS	\leftrightarrow / Out \rightarrow



To communicate the IP μ 8930 micro web server with pc or other standard RS-232 device, a level-shifting device must be used. The IP μ 8930 micro web server developer board has this chip built in, and routed to the RJ-45 serial connector.

There are many kinds of servers that can be used in our system, for example Micro Web Server and the Pico Web Server are the server categories that can be used in the system.

In this system we will use the IP μ 8930 Micro Web Server which is existence in our university lab, and which owns the required system needs.

The IP μ 8930 micro web server has a general purpose I/O pins (port A) which we use to communicate with other system components.

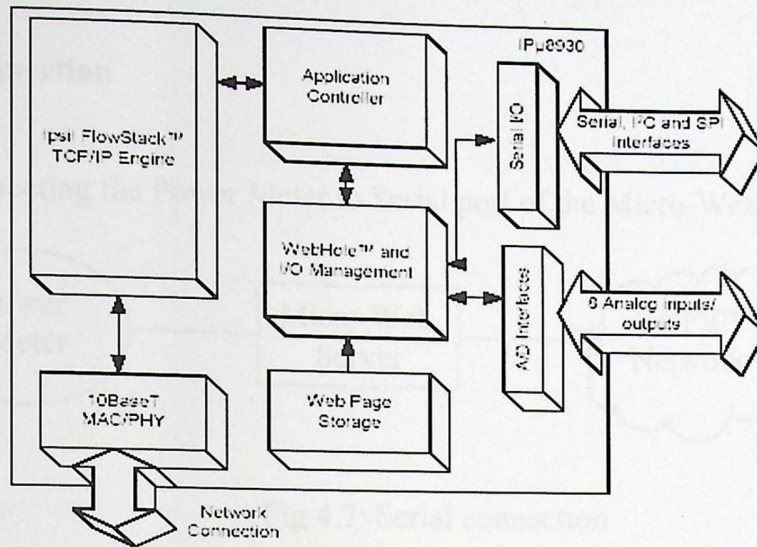


Fig 4.5: IPμ8930 Block Diagram

Types of possible connection to Power Meter

[A] Parallel Connection

1. Connecting the Power Meter to parallel port of the Micro-Web Server

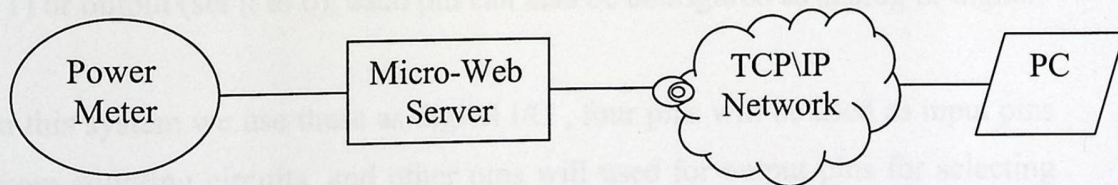


Fig 4.6: Parallel Connection

2. Writing a Java applet that uses the Ipsil Control Protocol (ICP) to read data from the Ipsil parallel port and process it in Java applet.
3. Uploading the Java applet to the Micro-Web Server system.
4. Embed the Java applet to a HTML page on the Micro-Web Server.

Whenever a user accesses the HTML page, the Java applet will run and get information to control the Power Meter using ICP.

[B] Serial Connection

1. Connecting the Power Meter to Serial port of the Micro-Web Server

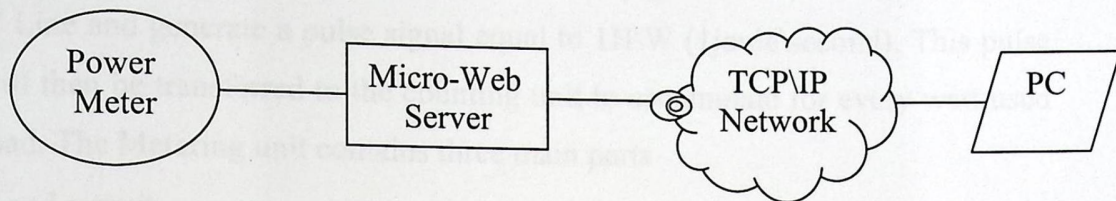


Fig 4.7: Serial connection

2. Writing a Java applet that uses the Ipsil Control Protocol (ICP) to read data from the Ipsil serial port and process it in Java applet.
3. Uploading the Java applet to the Micro-Web Server system.
4. Embed the Java applet to a HTML page on the Micro-Web Server.

Whenever a user accesses the HTML page, the Java applet will run and get information to control the Power Meter using ICP.

The IP μ 8930 micro web server contains 8 pins, each pin can be used for input (set it to 1) or output (set it to 0), each pin can also be configured as analog or digital.

In this system we use these as digital I/O, four pins will be used as input pins to read from counting circuits, and other pins will be used for output pins for selecting and controlling circuits.

To communicate the IP μ 8930 micro web server with PC or other standard RS-232 device, a level-shifting device must be used. The IP μ 8930 micro web server developer board has this chip built in, and routed to the RJ-45 serial connector.

In this project we will use two pins for control lines, one for counter selection, and four pins for KWH reading.

4.3 Metering units

The metering units consist of two meter units. This meter unit will be an implementation of previously designed circuit. Metering unit is used to be connected with AC Line and generate a pulse signal equal to 1Hz/W (1joule/second). This pulse signal will then be transferred to the counting unit to accumulate for every watt used by the load. The Metering unit contains three main parts

- Load circuit
- Optocouplers circuit
- Comparator circuit

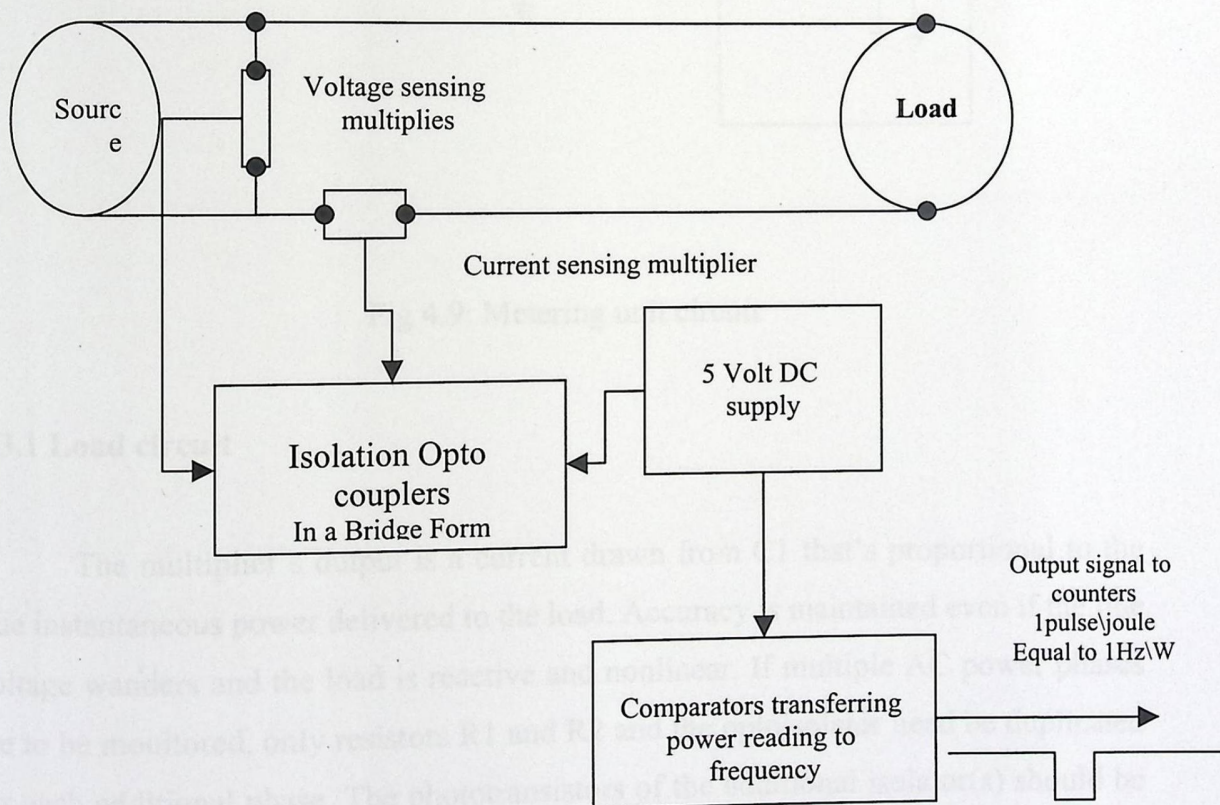


Fig 4.8: Block Diagram for meter circuit

We have chosen to build the meter because all its components are low cost and available.

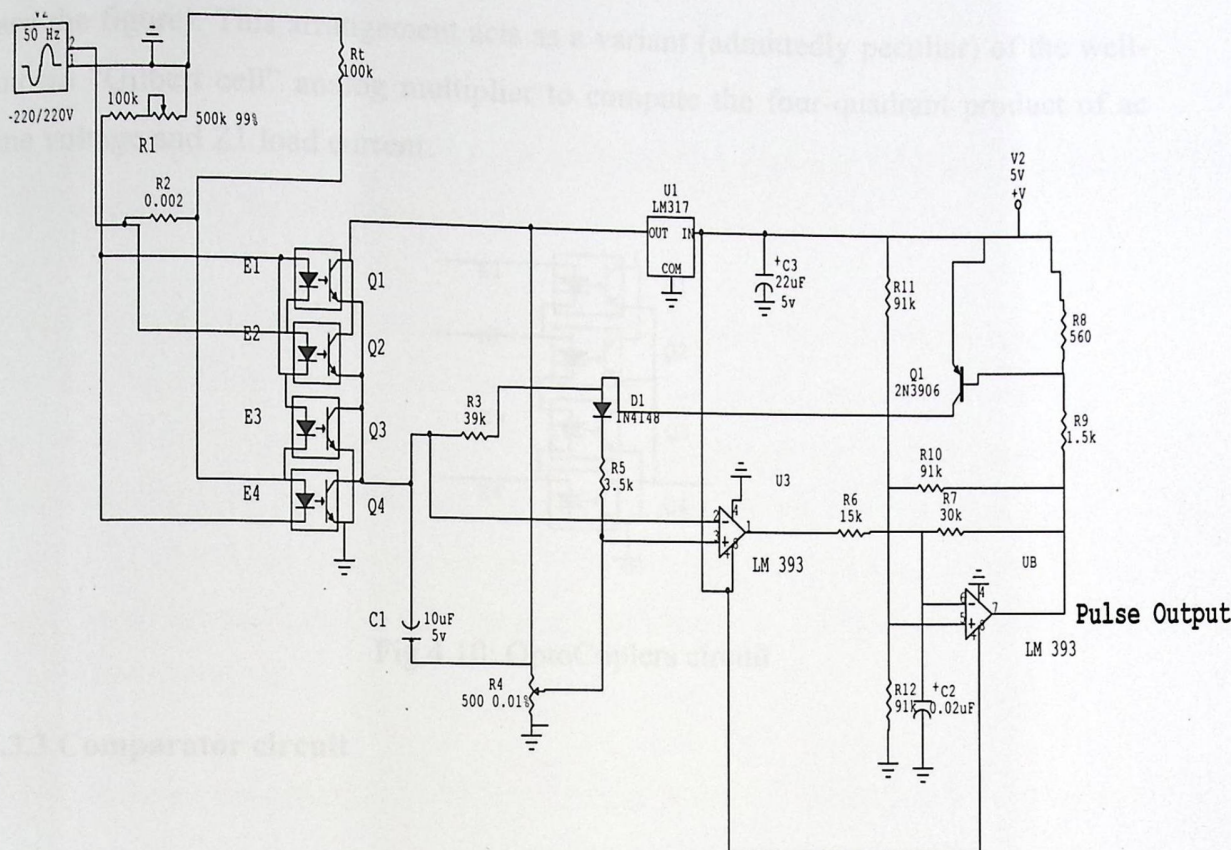


Fig 4.9: Metering unit circuit

4.3.1 Load circuit

The multiplier's output is a current drawn from C1 that's proportional to the true instantaneous power delivered to the load. Accuracy is maintained even if the line voltage wanders and the load is reactive and nonlinear. If multiple AC power phases are to be monitored, only resistors R1 and R2 and the optoisolator need be duplicated for each additional phase. The phototransistors of the additional isolator(s) should be paralleled with Q1-Q4.

4.3.2 Optocouplers circuit

The core of the wattmeter circuit is a quad-channel optoisolator consisting of LEDs E1-E4 and phototransistors Q1-Q4 connected in a double-bridge arrangement

(see the figure). This arrangement acts as a variant (admittedly peculiar) of the well-known "Gilbert cell" analog multiplier to compute the four-quadrant product of ac line voltage and Z1 load current.

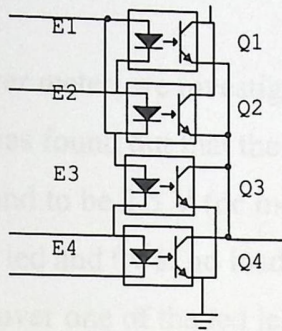


Fig 4.10: OptoCoplars circuit

4.3.3 Comparator circuit

The current-to-frequency converter formed by reference VR1, comparators A1 and A2, and associated discrete, converts the passively summed power-proportional currents to a 0-1200 Hz output appearing at A2 (pin 7).

4.3.4 Clock circuit

After we haven't obtained the 2 mohm resistor we decided to use elester A100 electricity meter for obtaining the clock pulse that corresponds to the electricity uses.

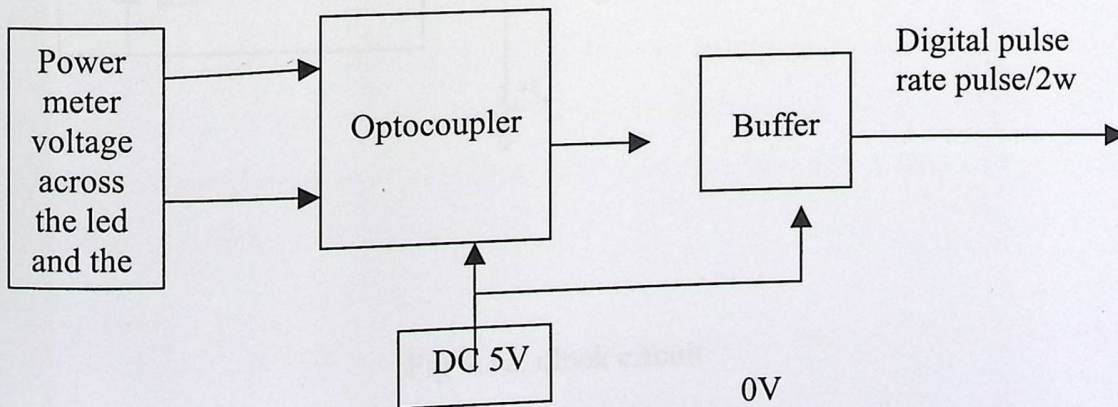


Fig 4.11: block clock circuit

The power (electricity) meter has a led that is turned on at a rate of 500pulse\KWH, which means that the led lights on and then switches off every 2W of energy usage. The lightening period is about 100 micro seconds. Our goal was to benefit from this pulsing so as to count the electricity usage.

In order to benefit from the power meter, we investigated the inside power meter circuit, using a multi-meter. It was found out that the pulsing led is connected to a resister, and the voltage was found to be 2.5 V (dc measurement) across the led and resistor during lightening of the led and 0v at no load. However this voltage was found to be ac and not dc, moreover one of the led legs was connected directly to the phase. So an ac measurement was taken and the result was 1.3v (ac) at lightening and 0v at led switched off.

During these investigations several circuits were tried. All of this was for the sake of a digital pulse from the power meter. Opto-couplers, inverters and phototransistor circuits were tested. Finally the following design worked correctly.

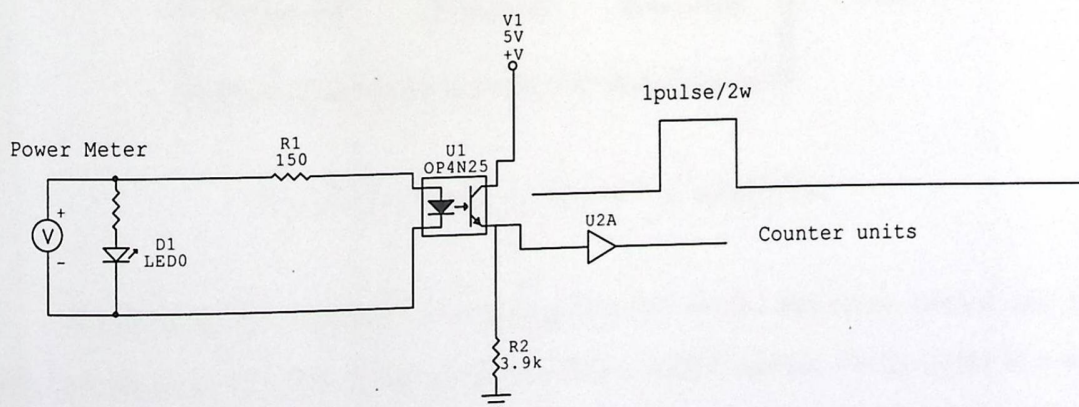


Fig 4.12: clock circuit

4.4 Counting unit

Counting unit circuit is used to count the pulses that are introduced by metering circuit; it contains four ripple counters as shown in Fig 4.9

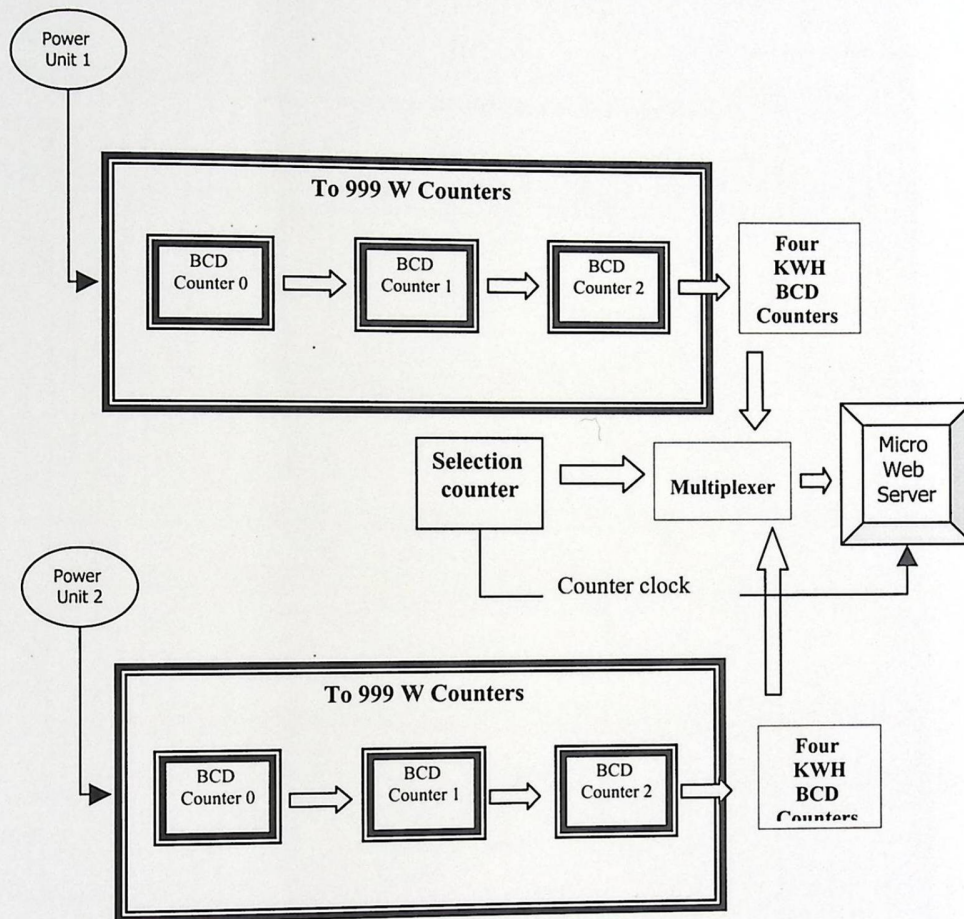


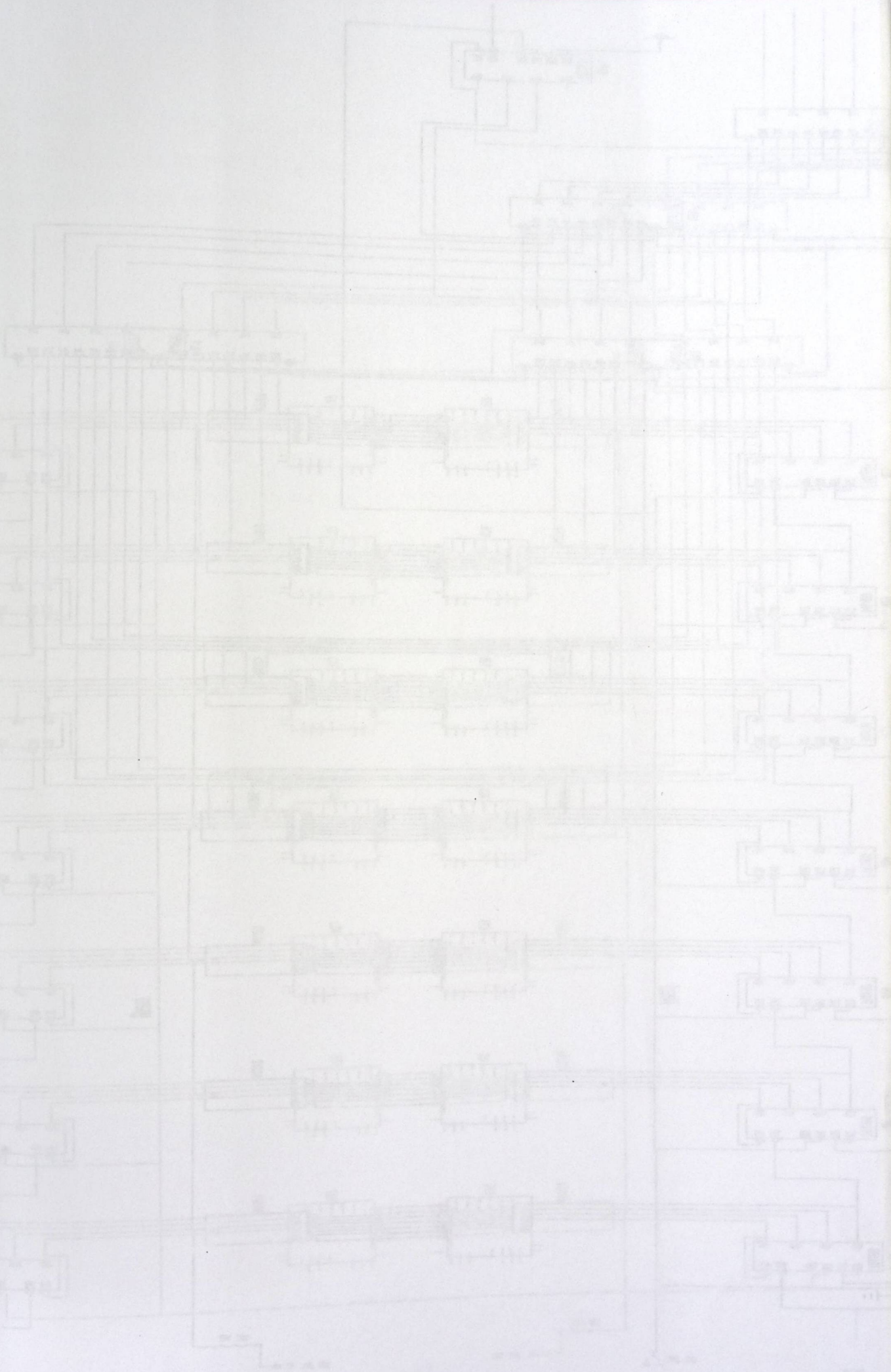
Fig 4.13: Counter units with a multiplexer

The first three counters connecting directly to the metering circuit and they counts the number of pulses that generated from meter circuit, every pulse is a watt, so these counters can count to 999 watt then increase the forth counter by one and turns to 000 to count a new kilo watt.

The fourth counter is used to count the number of kilo watt used, its output connected to a multiplexer which is connecting to the micro web server.

Multiplexer is used to enable which load micro web server will read.

Fig. 119. Computing unit with 1000 JFETs



Micro Web Server read fourth counter every unit of time determined by software by enabling the multiplexer then reset the counter by the reset bins which is connected to the micro web server

4.5 Controlling circuit

The controlling circuit consists of three relays that are controlled using the micro web server port A pins. The relays normally closed ends are put so as they close the lines and so the loads are normally connected to the ac power lines.

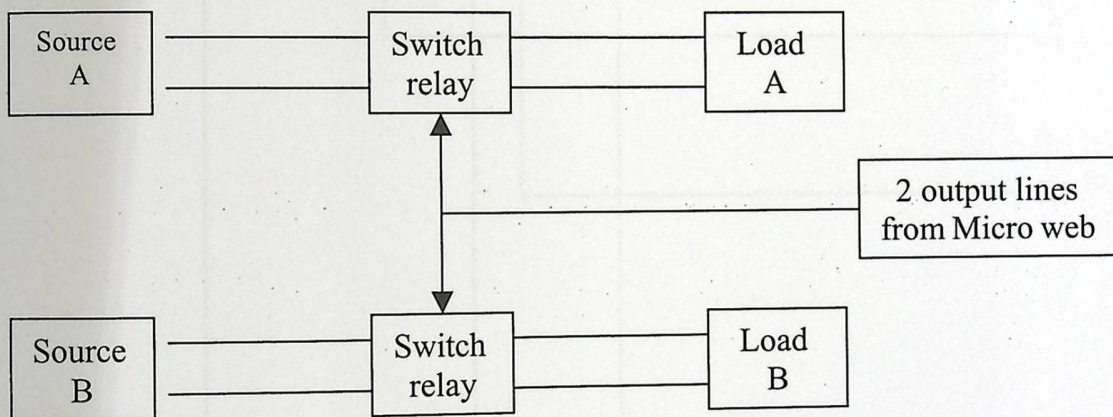


Fig 4.15: Block Diagram for Power Control Circuit

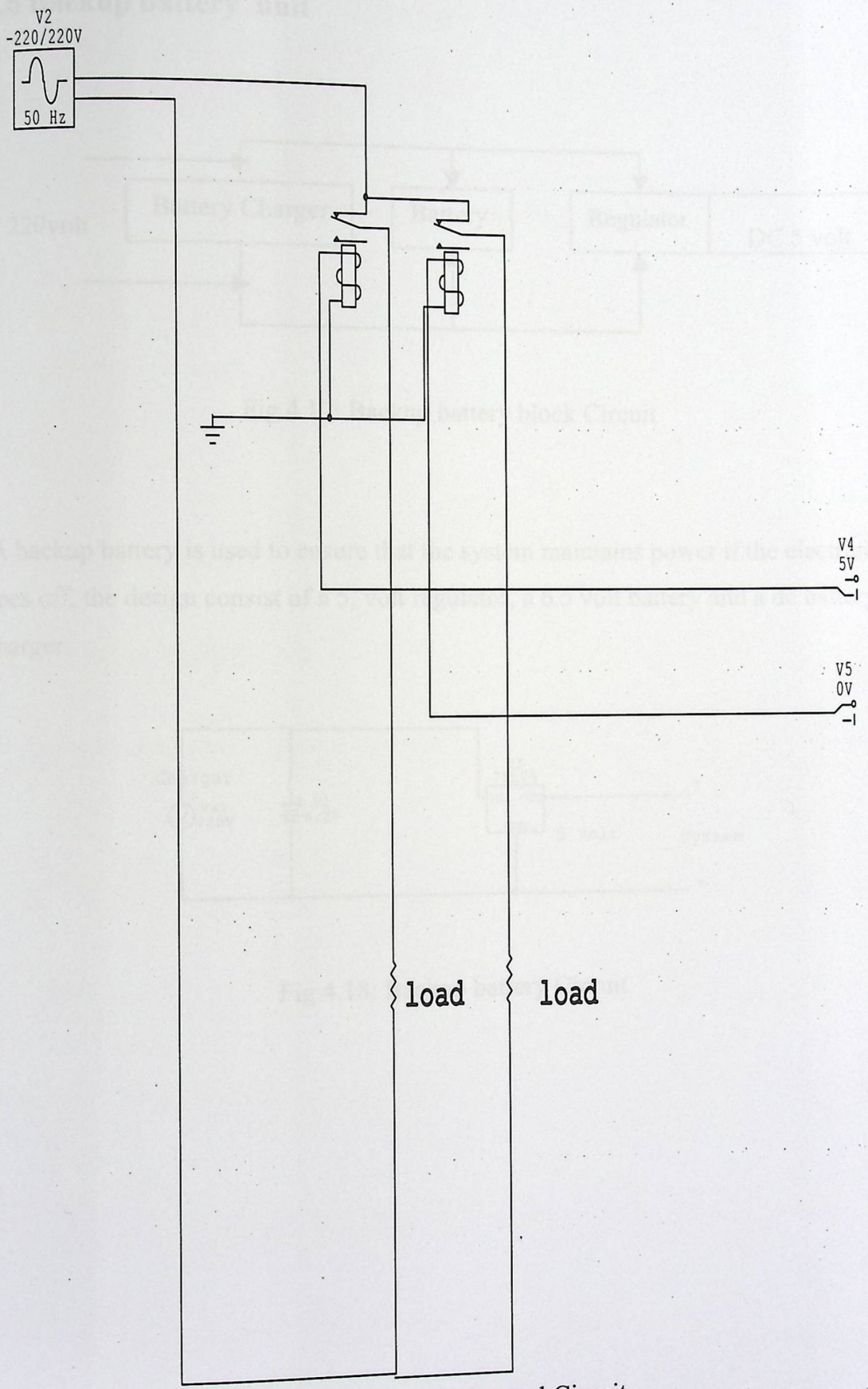


Fig 4.16: Power Control Circuit

4.6 Backup battery unit

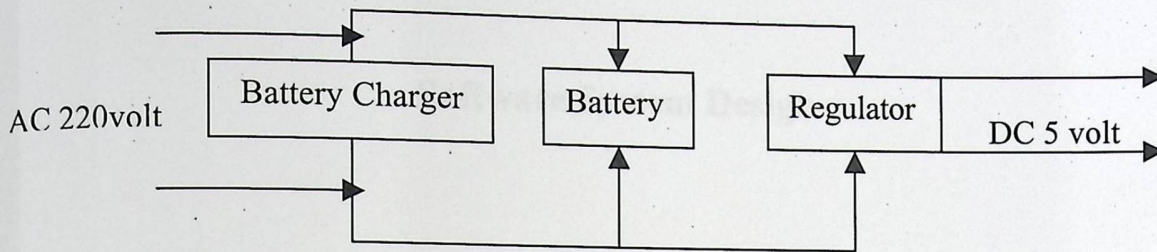


Fig 4.17: Backup battery block Circuit

A backup battery is used to ensure that the system maintains power if the electricity goes off, the design consist of a 5, volt regulator, a 6.5 volt battery and a dc battery charger.

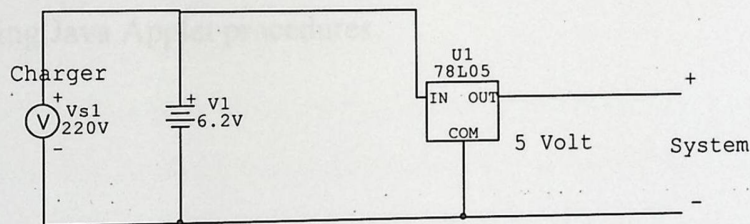


Fig 4.18: Backup battery Circuit

4.6 Backup battery unit

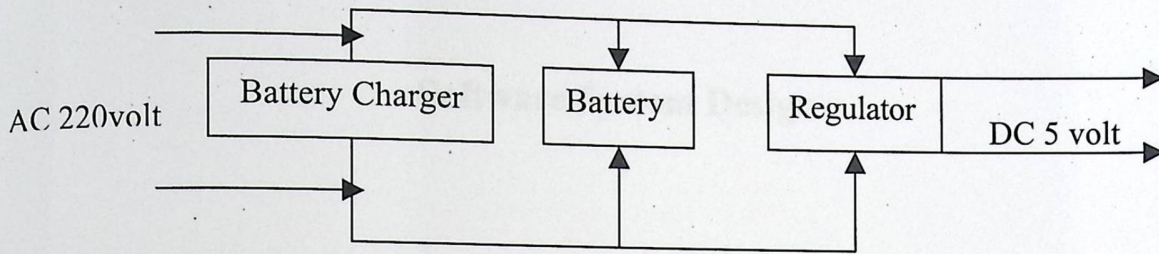


Fig 4.17: Backup battery block Circuit

A backup battery is used to ensure that the system maintains power if the electricity goes off, the design consist of a 5, volt regulator, a 6.5 volt battery and a dc battery charger.

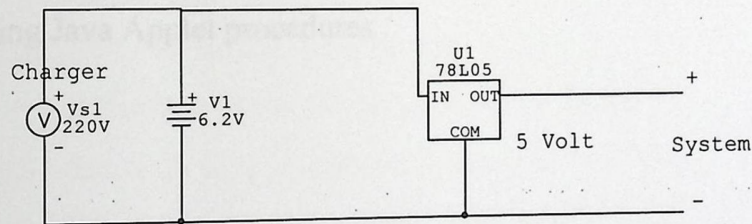


Fig 4.18: Backup battery Circuit

Chapter Five

Software System Design

5.1 Software Parameters

5.2 Software Tools

5.3 System Flowcharts

5.4 Web Pages Implementations

5.5 Implementing Java Applet procedures

1. Verifying the IP Address of the Micro-Web Server.
2. Port ID for the Socket path (Manufacturer to be 8080).
3. Path for Port Address to the Micro-Web Server.
4. Operation Code (OPCODE) of the operation to be performed through the Socket (Data Write).
5. Direction of Data (Data Received).
6. Command to be sent to the Power Meter.
7. Verifying the success of the specified operation.

Chapter Five

Software System Design

This chapter discusses the method for implementing the system. Software implementation will be in JAVA. And we will walk up with all different transition and behavior of system operation.

5.1 Software Parameters

Through programming of Micro-Web Server there is several steps that must be done related to ICP to the Micro-Web Server:

1. Verifying the IP Address of the Micro-Web Server.
2. Port ID for the Socket path (Manufacturer to be 8930).
3. Parallel Port Address in the Micro-Web Server.
4. Operation Code (OPCODE) of the operation to be performed through the Socket (Read/Write).
5. Direction of Data (Sent/Received).
6. Command to be sent to the Power Meter.
7. Verifying the success of the specified operation.

5.2 Software Tools

The following components from the software point of view are used:

1. Window XP SP2

To acts as Client.

2. Front Page 2003

In order to design the web pages of the system.

3. Java Creator Pro 2.00

In order to write Java Applet Codes.

4. JBuilder 2005 Enterprise

In order to design the interface of the Java Applet.

5. Java 2SDK 1.4.1

Contains the Frame work of the java environment.

6. Java Run Time Environment (JRE)

To support the Java Virtual Machine for the client processing state.

7. Ipsil Configuration Utility (ICU)

Software to upload files into the Micro-Web Server.

8. Power Meter Software

Configuration about Power Meters and there contents.

5.3 System Flowcharts

5.3.1 System Authorization

This flow chart shows how the user logging to the system, and how the system responds.

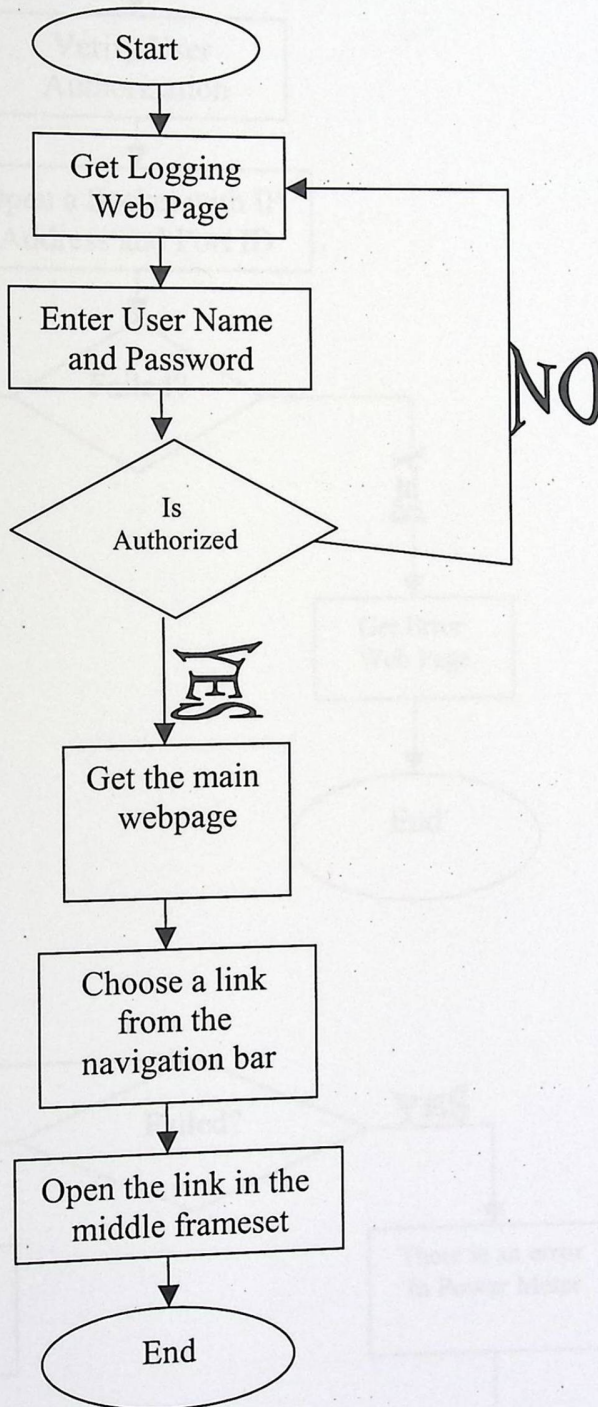


Fig 5-1: Power Meter Viewing

5.3.2 Power Meter Viewing

This flowchart shows how we can do several power meter options

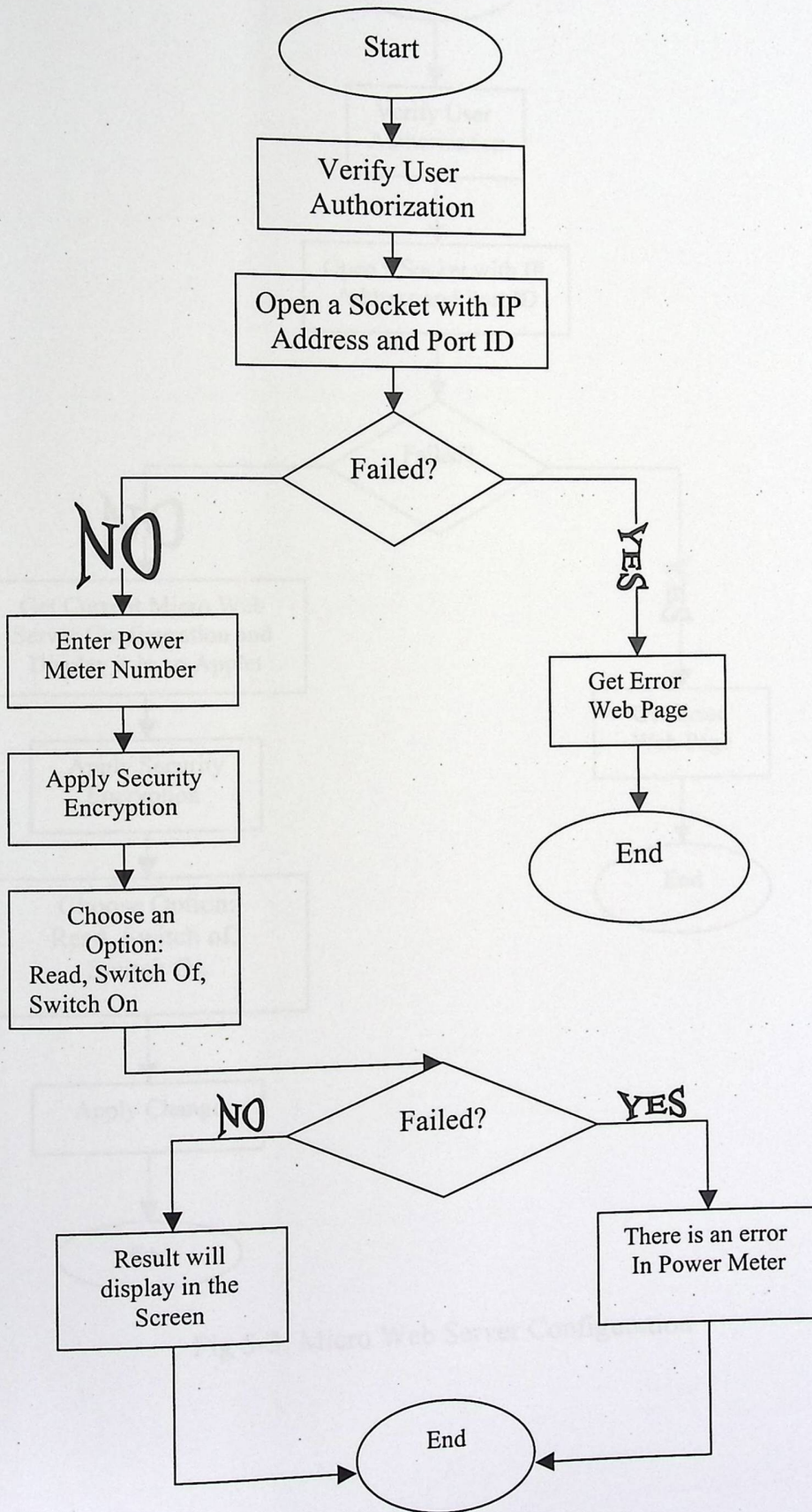


Fig 5-2: Power Meter Viewing

5.3.3 Micro-Web Server Configuration

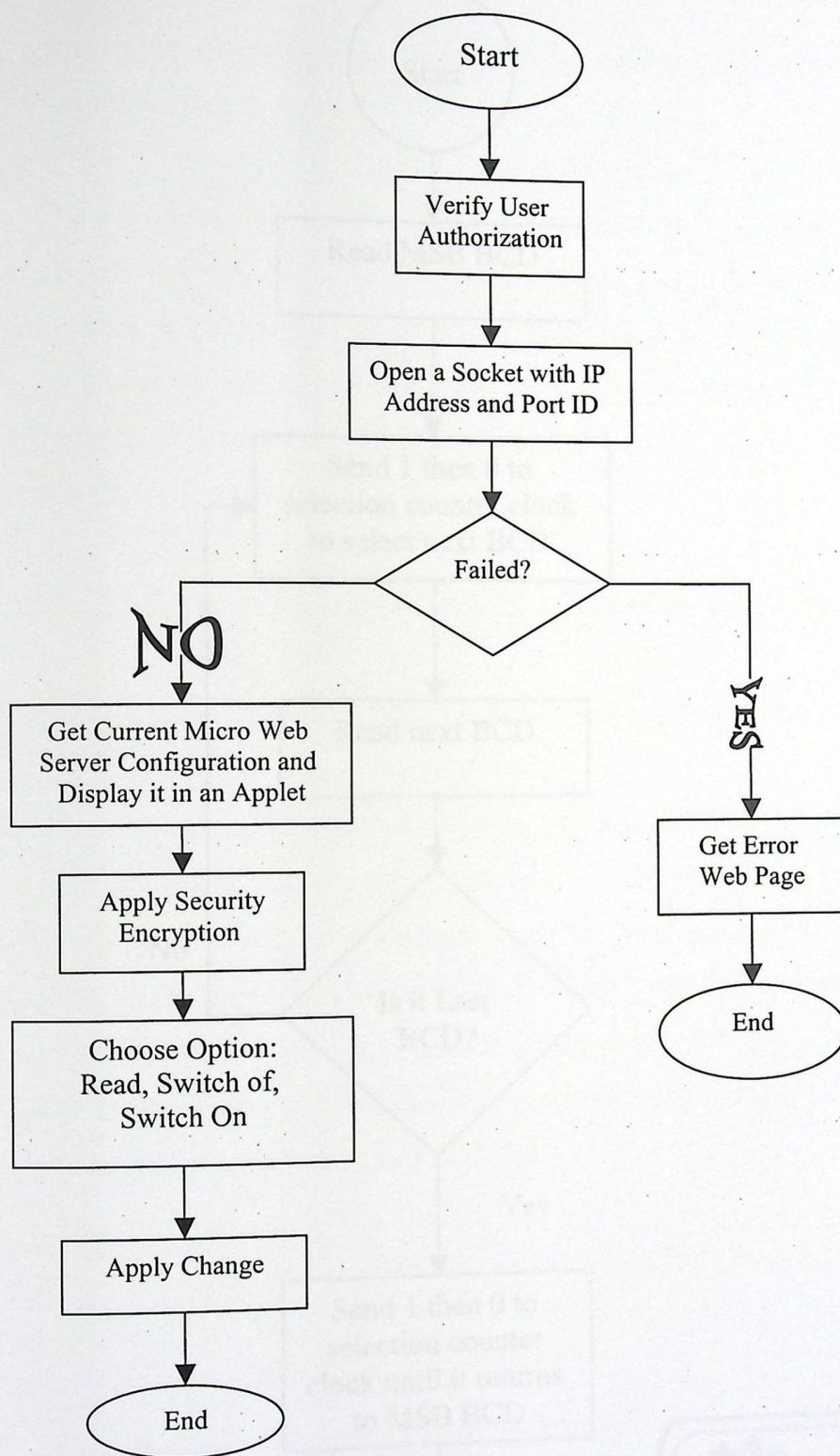


Fig 5-3: Micro Web Server Configuration

5.3.4 Read option

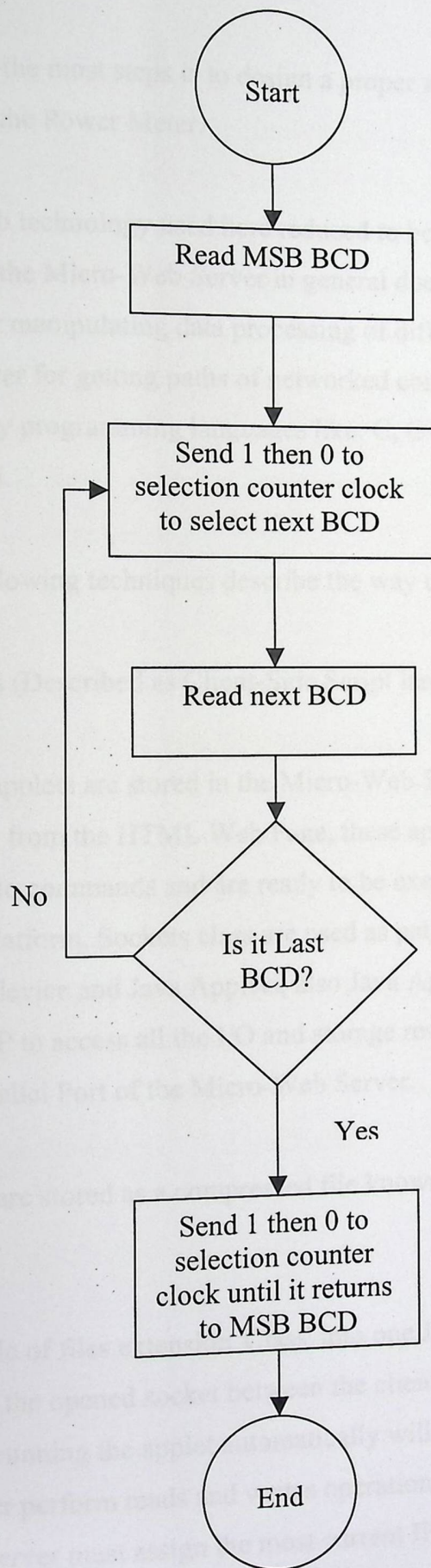
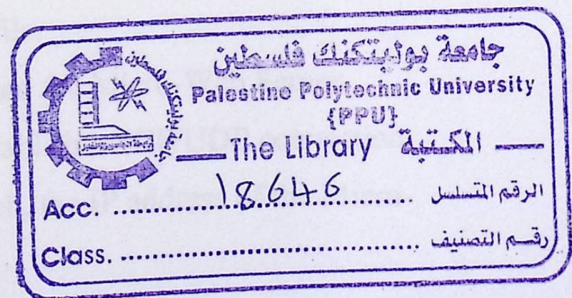


Fig 5-4: read option



5.4 Web Pages Implementations

One of the most steps is to design a proper web page because it acts as a face for controlling the Power Meter.

The web technology used here reduced to be Client-Side Scripting language; this is because the Micro-Web Server in general does not have the executive background for manipulating data processing of different languages. The Micro-Web Server is a server for getting paths of networked communications through an interface implemented by programming languages like: C, C++, Java Applets, LabView and Borland Delphi.

The following techniques describe the way used for implementing the system:

1. Java Applets (Described as Client-Side Script language)

These applets are stored in the Micro-Web Server External File System. They are come from the HTML Web Page, these applets contain several class files that divided into commands and are ready to be executed using Java Virtual Machine on the client platform. Sockets class are used as path in order to communicate between Ipsil device and Java Applets, also Java Applets used Ipsil Control Protocol (ICP) over TCP to access all the I/O and storage resources of the Micro-Web Server along with parallel Port of the Micro-Web Server.

These applets are stored as a compressed file known as 'Jar' file extension and include:

- Multiple of files extension 'Class' into one Jar file.
 - Secure the opened socket between the clients and the Micro-Web Server.
- When running the applet automatically will establish a TCP/UDP connection. In order perform reads and writes operation well, the IP address of the Micro-Web Server must assign the most current IP.

5.5 Implementing Java Applet procedures:

- Use JAR files instead of Java applets in Web Pages:

The IPm8930 file system requires the prefix of all filenames to be a three digit number (e.g., "160myfile", "170myfile"). However, class names for Java applets cannot begin with a number. The solution is to use just one JAR file that stores all of the java applets. A JAR file is a compressed archive of Java applets (or any other files) and can have a filename beginning with a number. The JAR file also has the added benefit of requiring less space on IPm8930 file system since it is compressed. For example, "203Classes.jar" is a valid file name and would be used if the JAR file was to be stored in file slot 203 on the IPm8930 file system. To use JAR files in web pages, use the option "ARCHIVE = [jar file name]" in your HTML code

- Padding JAR Files to 512 byte Multiples:

The IPm8930 file system clusters files into 512 byte blocks. When a Java JAR file is served to a client web browser, a data stream that is a multiple of 512 bytes is returned. However, if the original JAR file was not exactly a multiple of 512 bytes, there will be some extraneous trailing bytes (< 512 bytes) that are also sent to the client web browser. Some client browsers may block this JAR file as a potential virus because of these extraneous trailing bytes. The solution is to create JAR files that are exactly multiples of 512 bytes in size. This can be done easily by adding a uncompressed text file that contains as many spaces as needed to align the JAR file to a 512 byte multiple.

2. Java Script

Java Script uses codes that can be embedded in the HTML code of a web page to be executed when interacting with that page. It is the browser Plug-In software that takes on the execution of the codes.

The system rarely used Java Script to implement any web processing features.

Nevertheless, it is used to accomplish different issues like prohibit more than one user to access the system simultaneously, Check authorization for a user and generate friendly messages relevant to user's actions.

- Suggestions on Creating Web Pages on the IP μ 8930

The following are a list of suggestions for writing Web pages that will be stored on the Ipsil chip:

1. Any type of web object can be stored on the Ipsil chip including graphics, JAVA scripts, etc.—not just HTML.
2. When creating HTML files for the IP μ 8930, keep your design simple to minimize file size.
3. Also use whatever options are available on your favorite HTML editor to minimize unnecessary overhead. For example, Microsoft Word creates exceptionally large HTML files because it generates files which can be later re-edited by Microsoft Word and includes a significant amount of HTML code about the original Microsoft Word files to support this re-creation capability
4. File must be a minimum of 512 bytes long, and will be padded (with null characters) as necessary to end on a 512-byte boundary. This padding is done automatically when uploading using the ICU program.
5. it's strongly recommended that you name any files you create for uploading be named starting with the three-digit destination file number. For example, "075mypage.html". This helps you keep track of where files are loaded, and it also provides a way for the ICU to automatically determine where to save the file in the device.

5.6 System Interface

System Interface is the screen which the user can interact with, and so communicate with the system friendly. The followings are the system screens defined by the applet they load:

5.6.1 First page

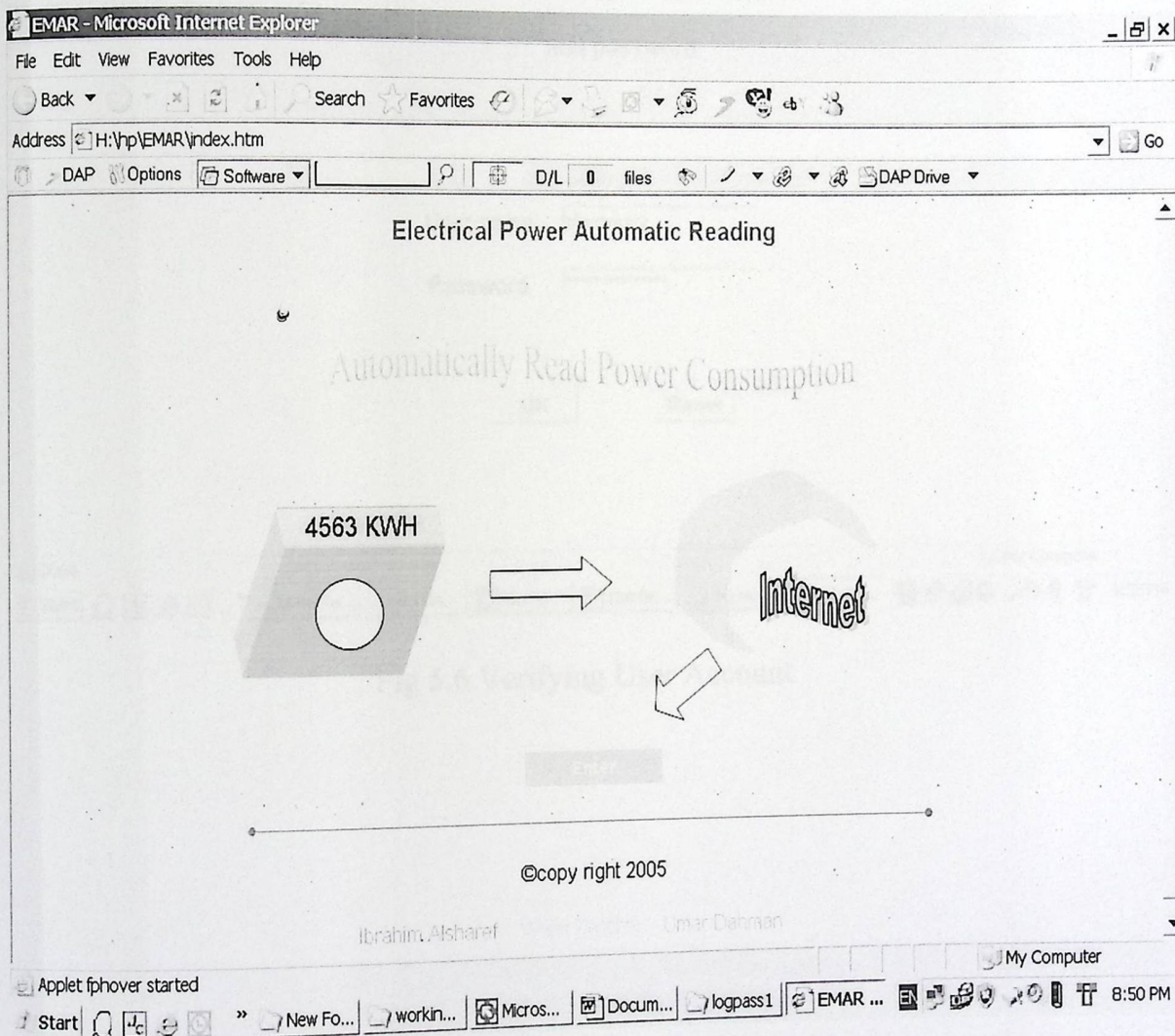


Fig 5.5 First Page

5.6.2 Verifying User Account

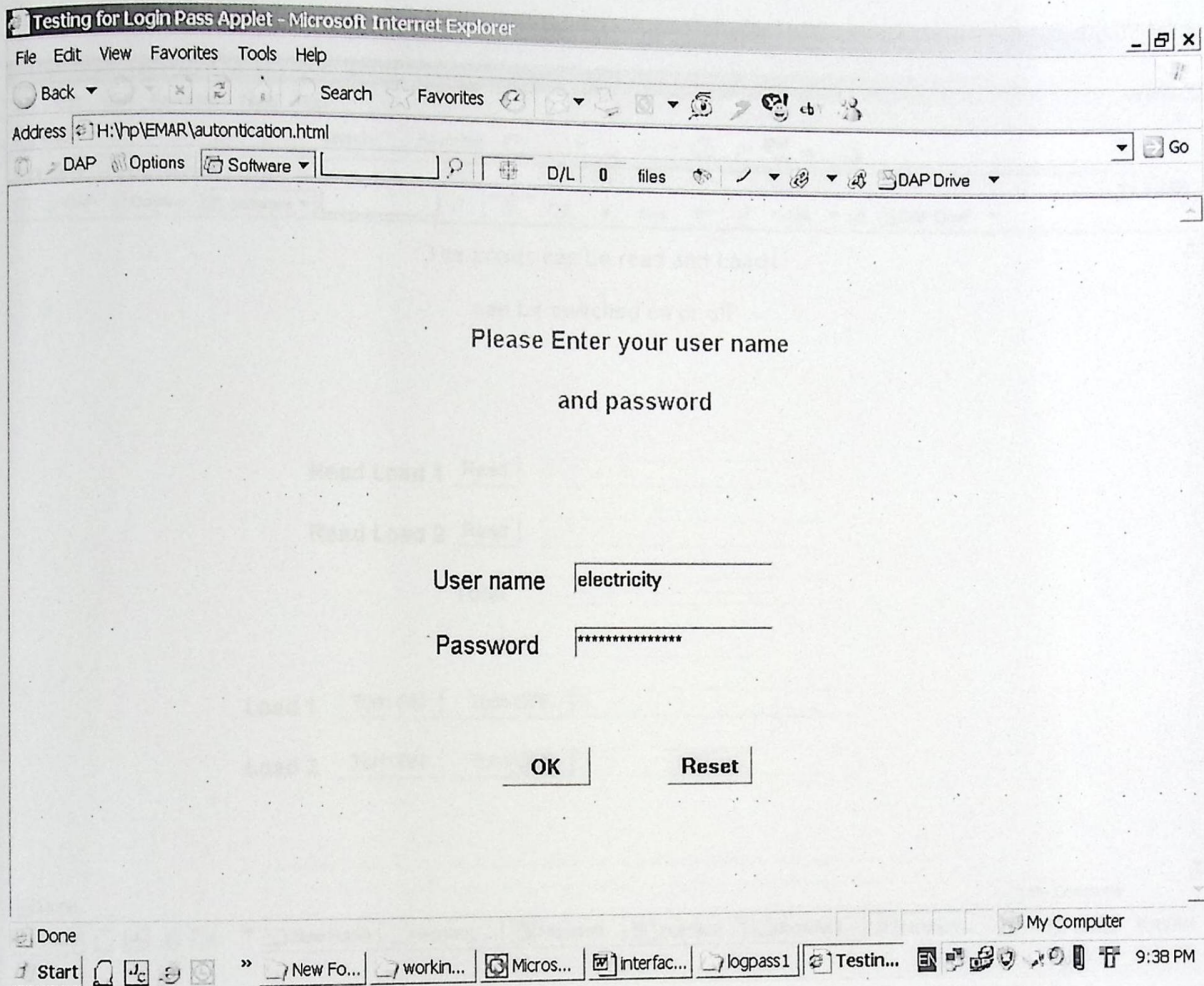


Fig 5.6 Verifying User Account

5.6.4 Power Meter Value

Chapter Six

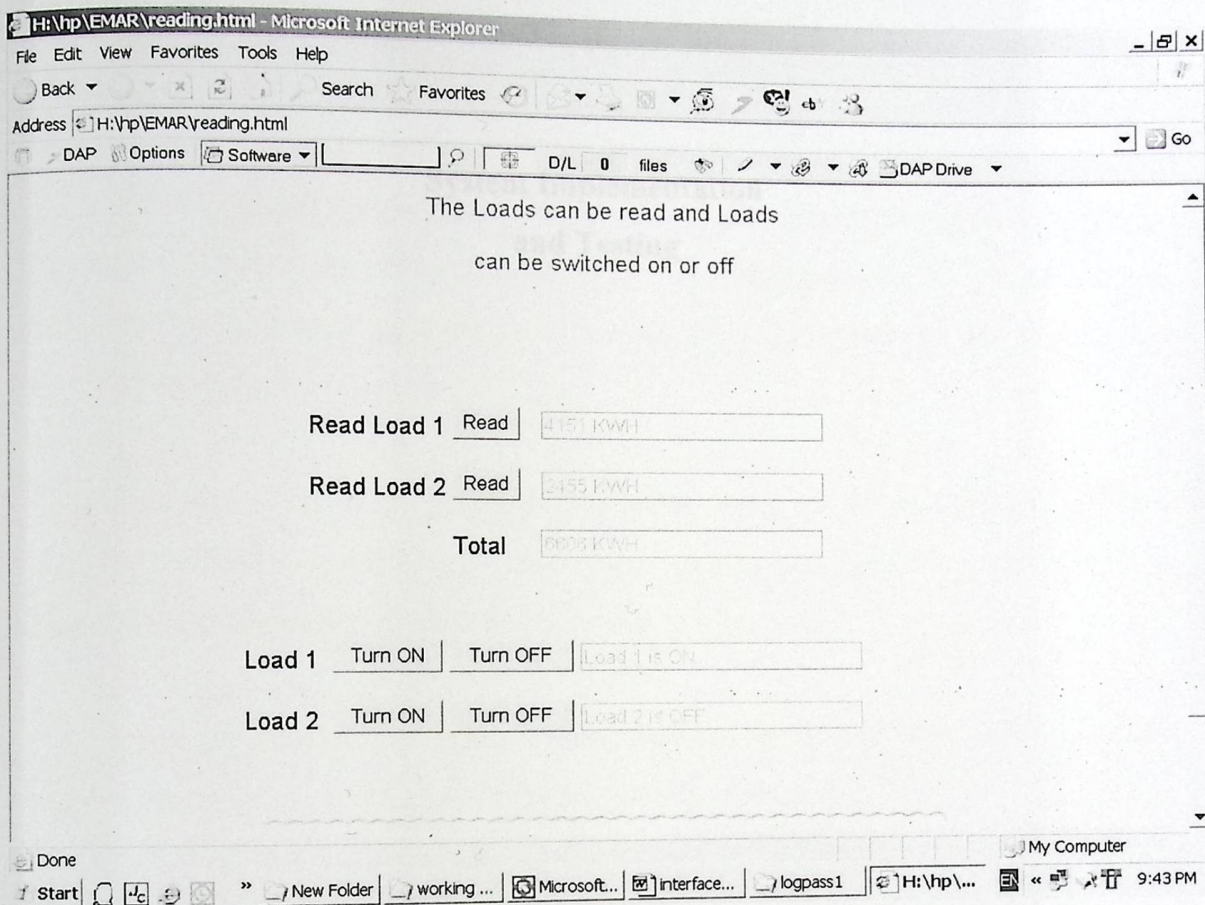


Fig 5.7 Power Meter Value

Chapter Six

Chapter Six

System Implementation and Testing

System Implementation and Testing

This chapter demonstrates the procedures used to test and evaluate the system operation and performance. System testing is an important and crucial step in implementing a system. It ensures the effective aspects of that system get before introducing it to its users.

This system has more than one issue to be tested. Some testing parts reflect a software, hardware or network of case. Also, testing procedures concentrate on a single device independent from the over whole system.

Here are the testing issues. They include the micro web server testing, hardware components testing and software application testing.

6.1 Introduction

The implementation and testing was done using the following tools and components:

- Connectors with different colors
- IC stands
- 30cm bread board
- All the IC's that are depicted in the design chapter (see chapter 4)
- Stripper tool for stripping the connectors on the IC's leads
- A wire grabber and a wire cutter
- A digital Millimeter for testing

A phase's approach was carried for implementing and testing the hardware components. Then the spoken and web pages were implemented and tested. The implementation and testing was for one load. More than one system software source code is included in the appendix.

Chapter Six

System Implementation and Testing

This chapter demonstrates the methods and procedures used to test and examine the system operation and behavior. System testing is an important and crucial step in implementing a system. It senses the effectiveness of that system just before introducing it to its users.

This system has more than one issue to be tested. Some testing parts reflect a software, hardware or networked case. Also, testing procedures concentrate on a single device independent from the over whole system.

Here are the testing issues. They include the micro web server testing, hardware components testing and software applets testing.

6.1 Introduction

The implementation and testing was done using the following tools and components:

- Connectors with different colors
- IC stands
- 50 cm broad board
- All the ICs that are depicted in the design chapter (see chapter 4)
- Wrapper tool for wrapping the connectors on the ICs stands
- A wire grabber and a wire cutter
- A digital Millimeter for testing

A phases approach was carried for implementing and testing the hardware components. Then the applets and web pages were implemented and tested. The implementation and testing was for one load. More over the system software source code is included in the appendix.

6.1 Testing the Micro Web Server

Testing the Micro Web Server involves doing more than one step. In order to test it, there must be a simple installed network; this is accomplished by connecting the Micro Web Server with a simple client (PC) that has an Internet Browser (i.e. Internet Explorer, or Netscape navigator). Testing the Micro web server is done by simply asserting one of the following conditions:

- Structuring a cross over cable between a PC and the Micro Web Server
- Connecting the Micro Web Sever to one of a Hub ports

After connecting the cable between the Micro Web Server and the PC, power on the Micro Web Server to complete the network, some testing commands are done:

1. Testing the TCP/IP Protocol

This test is configured at the client side PC because the Micro Web Server already supports the TCP and UDP Protocols. The test is done simply by writing the following command on the Command Line Shell:

```
Ping 127.0.0.1
```

The following reply indicates the success of the installed TCP/IP protocol:

```
Pinging 127.0.0.1 with 32 bytes of data:
```

```
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

```
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

```
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

```
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

```
Ping statistics for 127.0.0.1:
```

```
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Testing Micro Web Server Reachability

The Micro Web Server when connected to a network is configured as a DHCP client. Detection packets are sent when sharing a network to have an IP Address from a DHCP server. When no server satisfies that, the 192.168.0.254 IP address is assigned to the Micro Web Server.

Testing the Micro Web Server assumes no DHCP server is found. So, the following command will test the Micro Web Server:

```
Ping 192.168.0.254
```

The following reply indicates the success of Micro Web Server accessibility:

```
Pinging 192.168.0.254 with 32 bytes of data:
```

```
Reply from 192.168.0.254: bytes=32 time=1ms TTL=128
```

```
Reply from 192.168.0.254: bytes=32 time=1ms TTL=128
```

```
Reply from 192.168.0.254: bytes=32 time=1ms TTL=128
```

```
Reply from 192.168.0.254: bytes=32 time=1ms TTL=128
```

```
Ping statistics for 192.168.0.254:
```

```
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

```
Approximate round trip times in milli-seconds:
```

```
Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

6.2 Testing TCP/IP Sockets

Testing win sockets needs applying some codes and executing it through a java applet. There are two main applets used to test both the Serial Port and the 8 General I/O pins.

6.2.1 Testing Port A

There are 8 general purpose I/O pins installed on the Micro Web Server. The following testing code is used to implement a socket through a java applet to flash 8 LEDs ON and OFF:

```
import java.applet.Applet;
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;

public class setDigitalHi extends Applet
{
private int IPSIL_PORT = 8930;
InetAddress ia = null;
Socket sock = null;
InputStream sIn = null;
OutputStream sOut = null;
StringBuffer buffer = new StringBuffer();
byte [] setAllDigitalOutput =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
0x06, // "Write EEPROM" cmd (1 byte len)
0x00, 0x00, 0x01, 0x13, // Start location (4 bytes len)
(byte)0x86, // All channels Digital (1 byte len)
0x00 // All channels as Outputs (1 byte len)
};

byte [] flashLedsOn =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
0x06, // "Write EEPROM" cmd (1 byte len)
0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
(byte)0xff // All channels HI (1 byte len)
};
```



```

byte [] flashLedsOff =
{ 0x00, 0x00, 0x00, 0x00, 0x00,    // Password   (5 bytes len)
0x06,    // "Write EEPROM" cmd   (1 byte len)
0x00, 0x00, 0x01, 0x17,    // Start location   (4 bytes len)
(byte)0x00    // All channels Low   (1 byte len)
};

byte [] resp = new byte[1024];    // Network response buffer

public void init() {
try {
ia = InetAddress.getByName( getCodeBase().getHost() );
sock = new Socket(ia, IPSIL_PORT);
addItem(ia.toString());
sIn = sock.getInputStream();
sOut = sock.getOutputStream();
}
catch (Exception e)
{
    e.printStackTrace();
    addItem(" Init " );
}
}

private void addItem(String newWord)
{
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

public void paint(Graphics g)
{
    g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);
    g.drawString(buffer.toString(), 5, 15);
}

public void start() {

```

```

try{
    sOut.write(setAllDigitalOutput);
    sIn.read(resp);
    sOut.write(flashLedsOn);
    sIn.read (resp);
    Thread.sleep( 1000 );
    addItem("LEDs ON!");
    sOut.write(flashLedsOff);
    sIn.read (resp);
    Thread.sleep( 1000 );
    addItem(" LEDs OFF!");
}
catch (Exception e)
{
    e.printStackTrace();
    addItem(" Start ");
}
}
}
}

```

6.2.2 Testing Parallel Port

The testing of the parallel port via a socket is implemented by connecting a LED to the transmit line of the parallel port. Then an applet is written to flash the led ON and OFF, here is the testing applet code:

```

//an applet to turn port A to 1 in digital
import java.applet.Applet;
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;

```

```

public class setDigitalHi extends Applet
{
    // Constants
    private int IPSIL_PORT = 8930;           // Ipsil device port number

    // Variables
    InetAddress ia = null;
    Socket sock = null;                     // Socket
    InputStream sIn = null;                 // Input data stream
    OutputStream sOut = null;               // Output data stream instance

    StringBuffer buffer = new StringBuffer();

    // Arrays
    // Write RAM variable ICP command
    // Configures all channels as Digital Outputs
    // Operation code = 0x06, address in RAM 0x113, bytes to write 2
    byte [] setAllDigitalOutput =
    { 0x00, 0x00, 0x00, 0x00, 0x00,          // Password (5 bytes
len)
        0x06,                               // "Write EEPROM" cmd (1 byte len)
        0x00, 0x00, 0x01, 0x13,             // Start location (4 bytes len)
        (byte)0x86,                          // All channels Digital (1 byte len)
        0x00,                               // All channels as Outputs (1 byte len)
    };

    // Set all digital outputs to HI level
    // Write RAM variable ICP command
    // Operation code = 0x06, address in RAM 0x117, 1111 1111 - all channels "1" output
    // Each bit of the last byte represents each channel state accordinly
    byte [] setAllDigitalHi =
    { 0x00, 0x00, 0x00, 0x00, 0x00,          // Password (5 bytes
len)

```

```

    0x06,          // "Write EEPROM" cmd    (1 byte len)
    0x00, 0x00, 0x01, 0x17, // Start location          (4 bytes
len)
    (byte)0xff,    // All channels HI        (1 byte len)
};

```

```

byte [] resp = new byte[1024]; // Network response buffer

```

```

public void init()

```

```

{

```

```

    try

```

```

    {

```

```

        //ia = InetAddress.getByName("192.168.1.101");

```

```

        //ia = InetAddress.getLocalHost();

```

```

        //addItem(ia.getHostAddress());

```

```

        ia = InetAddress.getByName( getCodeBase().getHost() );

```

```

        // Init TCP socket. Connect to the device itself on Ipsil port

```

```

        sock = new Socket(ia, IPSIL_PORT);

```

```

        addItem(ia.toString());

```

```

        // Create input data stream

```

```

        sIn = sock.getInputStream();

```

```

        // Create output data stream

```

```

        sOut = sock.getOutputStream();

```

```

    }

```

```

catch (Exception e)

```

```

{

```

```

    e.printStackTrace();

```

```

    addItem(" Init ");

```

```

    }
}

private void addItem(String newWord)
{
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

public void paint(Graphics g)
{
    //Draw a Rectangle around the applet's display area.
    g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);

    //Draw the current string inside the rectangle.
    g.drawString(buffer.toString(), 5, 15);
}

public void start()
{
    try
    {
        // Send ICP command - Configure all channels as digital outputs
        sOut.write(setAllDigitalOutput);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        // Send ICP command - Set HI all on all of the channels
        sOut.write(setAllDigitalHi);
    }
}

```

```

// Read and ignore replay byte from the Ipsil device
sIn.read (resp);

addItem(" All Digital HI Done!");
}

catch (Exception e)
{
    e.printStackTrace();
    addItem(" Start ");
}
}

} //end of the applet

```

//an applet to turn port A to 0 in digital

```
import java.applet.Applet;
```

```
import java.io.*;
```

```
import java.net.*;
```

```
import java.util.*;
```

```
import java.awt.*;
```

```
public class setDigitalLo extends Applet
```

```
{
```

```
    // Constants
```

```
    private int IPSIL_PORT = 8930;           // Ipsil device port number
```

```
    // Variables
```

```
    InetAddress ia = null;
```

```
    Socket sock = null;           // Socket
```

```
    InputStream sIn = null;       // Input data stream
```

```
    OutputStream sOut = null;     // Output data stream instance
```

```

StringBuffer buffer = new StringBuffer();

// Arrays
// Write RAM variable ICP command
// Configures all channels as Digital Outputs
// Operation code = 0x06, address in RAM 0x113, bytes to write 2
byte [] setAllDigitalOutput =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes
len)
    0x06, // "Write EEPROM" cmd (1 byte len)
    0x00, 0x00, 0x01, 0x13, // Start location (4 bytes
len)
    (byte)0x86, // All channels Digital (1 byte len)
    0x00, // All channels as Outputs (1 byte len)
};

// Set all digital outputs to LO level
// Write RAM variable ICP command
// Operation code = 0x06, address in RAM 0x117, 0000 0000 - all channels "1" output
// Each bit of the last byte represents each channel state accordingly
byte [] setAllDigitalLo =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
    0x06, // "Write EEPROM" cmd (1 byte len)
    0x00, 0x00, 0x01, 0x17, // Start location (4 bytes
len)
    (byte)0x00, // All channels LO (1 byte len)
};

byte [] resp = new byte[1024]; // Network response buffer

public void init()
{

```

```

try
{
    //Draw a rectangle around the applets display area
    //ia = InetAddress.getByName("10.1.1.20");
    //ia = InetAddress.getLocalHost();
    //addItem(ia.getHostAddress());
    ia = InetAddress.getByName( getCodeBase().getHost() );

    // Init TCP socket. Connect to the device itself on Ipsil port
    sock = new Socket(ia, IPSIL_PORT);
    addItem(ia.toString());

    // Create input data stream
    sIn = sock.getInputStream();

    // Create output data stream
    sOut = sock.getOutputStream();

}

catch (Exception e)
{
    e.printStackTrace();
    addItem(" Init " );
}
}

```

```

private void addItem(String newWord)
{
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

```



```

public void paint(Graphics g)
{
    //Draw a Rectangle around the applet's display area.
    g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);

    //Draw the current string inside the rectangle.
    g.drawString(buffer.toString(), 5, 15);
}

public void start()
{
    try
    {
        // Send ICP command - Configure all channels as digital outputs
        sOut.write(setAllDigitalOutput);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        // Send ICP command - Set HI all on all of the channels
        sOut.write(setAllDigitalLo);

        // Read and ignore replay byte from the Ipsil device
        sIn.read (resp);

        addItem(" All Digital Lo Done!");
    }

    catch (Exception e)
    {
        e.printStackTrace();
        addItem(" Start ");
    }
}

```

}

} //end of the applet

6.4 Implementing and testing projects hardware components

General hardware components testing:

BCD counter: a simple circuit was designed for testing the BCD counter, which was 74LS90, and the results were the BCD sequence (0, 1, 2 9, 0)

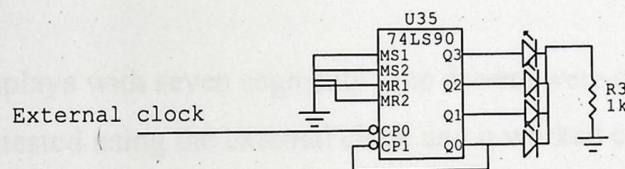


Fig 6.1 BCD counter testing

Multiplexer: two BCD counters with a quadruple multiplexer was tested, a selection line was used to select between the two counters. The result was that it worked correctly as desired.

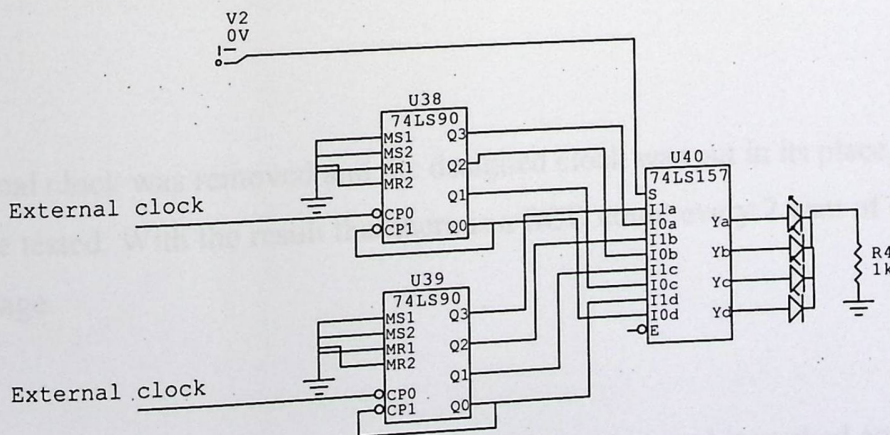


Fig 6.2 Quad 2-1 Multiplexer testing

Testing phases:

Phase 1

The clock circuit was implemented and tested separately. And it counted correctly, giving a pulse every 2 watt of energy consumption.

Phase 2

Several BCD counters were connected in series to form a counter from (0000000) to (9999999), an external clock was chosen to drive these counters.

Phase 3

Seven segments displays with seven segments Line drivers were connected with BCD counters, and were tested using the external clock and it worked correctly.

Phase 4

3 multiplexers were added to the circuit in addition to a BCD counter for selection from the 4 most significant counters. All the components of this phase and the previous phases were tested and the results were correct.

Phase 5

The external clock was removed and the designed clock was put in its place, and all of these were tested. With the result that there is a BCD count every 2 watt of electrical energy usage.

Phase 6

The control circuit was implemented using a switch relay, and it worked correctly switching on and off the led, that was chosen instead of the 220 ac line for testing. This led will be replaced by the 220 v ac line in the final phase.

Phase 7

Finally the led was replaced by the 220 v ac line, and the relay with all BCD interface from the multiplexer and the selection line were connected to the micro web server and tested using web pages and java applets.

6.5 Testing Java Applets

Testing Java Applets is one of the most crucial phases of testing. This fact stems from that java applets along with the web pages contains them is the interface of the system at which a user can interact.

The following steps complete applets testing on the web pages stored in the Micro Web Server:

1. Transferring the applet's HTML and JAR files to the Micro Web Server file system
 2. Using IPSil Configuration Utility (ICU) to accomplish this step
 3. Ensuring that the HTML and JAR file locations match the CODEBASE attribute in the HTML file and that the CODE attribute has the fully qualified class name
- Testing the applet in various browsers. If the applet fails to load, check that the browser is Java-enabled. Also check the browser's Java Console for error messages.

Testing Java Applets Buttons

1) Read Button

We attached the power meter to light and make it consumes power, then after that we pressed read button and the reading result appears into screen multiply by two.

2) Select Button

Select button is used to determine which counter we will read from, but we concern into the four counters.

3) Turn Off

This button is used to switch off the electricity by sending a signal of five volt to the relay.

First we will read the value of the power meter, and then we will determine whether it needs to turn off or not

We attached a light, and by this button we switched off the electricity for.

4) Turn On

This button is used to switch on the electricity. In the normal mode it will be on.

Chapter 7

Conclusion and future work

Project conclusions

Conclusion and future work

- We have built the counter circuit together with multipliers.
- The clock circuit implemented and connected to both the power meter and the clock circuit.
- A switch relay was used to build the control circuit.
- A 5.2 volt battery with a 5 volt regulator was used to compensate for electricity cut-off.
- All of the above were tested together using the software (the java applet and the web pages). The system worked correctly.

Project problems

- Electricity cut off during the project's work.
- There is a lack of strong practical experience in the developing team.
- There is a shortage of electrical components in local stores.

Future work:

- The system can be implemented using a different interface and a different transmission media such as serial interface and wireless media, or infrared interface and electrical media.
- The system can be extended to measure water, gas and other things.

An advanced management system can be built to

Chapter 7

Conclusion and future work

Project conclusions

- We have built the counter circuit together with multiplexer.
- The clock circuit implemented and connected to both the power meter and the clock circuit.
- A switch relay was used to build the control circuit.
- A 6.2 volt battery with a 5 volt regulator was used to compensate for electricity cut off.
- All of the above were tested together using the software (the java applets and the web pages). The system worked correctly.

Project problems

- Electricity cut off during the project's work.
- There is a lack of strong practical experience in the developing team.
- There is a shortage of electrical components in local stores.

Future work:

- The system can be implemented using a different interface and a different transmission media such as serial interface and wireless media, or infrared interface and electrical media.
- The system can be extended to measure water, gas and other things.

An advanced management system can be built to

Appendices

Appendix A: Data Sheets

Appendix B: IPU8930 Schematic Diagrams

Appendix C: Project ICs Data Sheets

Appendix D: Project software source code Data Sheets

Appendix A

Several tables to describe micro web server operations

Data Sheets

Field	Length (bits)	Field Name	Description
0-31	32	PASSWORD	16-bit password for ASCII command - always required. Will be returned if "None" (ASCII 0) is not provided. If ASCII 0 is provided it is ignored and the field is zero.
32-39	8	ADDR	Address argument for the above operations. Addresses are in the "short" format, e.g., 10000000. 10 = FC hex (16.1), 11 = FF hex (17.1).
40-47	8	PORT	Number of bytes to read. Binary number - may be from 1-32 for ICP packets and 1-20 for TCP packets.

Table 3-1: HTTP Packet - Read Request

Field	Length (bits)	Field Name	Description
0-03	4	ADDR	Address argument for the requested read. Same address as that in the read request.
04-07	4	INSTAMP	INST is inst. 150 = inst. This is the running count of time since last power up increments in 64-bit increments. These bits can be used to attach a time reference to the data.
08-31	24	RESPDATA	Requested data.

Table 3-2: ICP Packet - Read Response

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password—always required. Will be ignored if secure mode (SECMODE) is not enabled. If SECMODE is disabled, it is recommended that this field be zero padded.
0x05	1	OPCODE	2 = Read from External EEPROM 4 = Read from Internal EEPROM 7 = Read from RAM location 9 = Read from serial port 10 = I ² C Read (v1.1) 12 = SPI Read (v1.1)
0x06	4	ADDR	Address argument for the above operations. Addresses are in 'big-endian' format, e.g., MSB comes first.
0x0A	1	READLEN	Number of bytes to read. Binary number—must be from 1-32 for UDP packets and 1-20 for TCP packets

Table 5-1: ICP Packet – Read Request

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	4	ADDR	Address argument for the requested read. Same address as sent in the read request
0x04	4	TIMESTAMP	MSB is first; LSB is last. This is the running count of time since last power up measured in 8 ms increments. These bytes can be used to attached a time reference to the data
0x08	1-32	READDATA	Requested data

Table 5-2: ICP Packet – Read Response

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password—always required. Will be ignored if secure mode (SECMODE) is not enabled. If SECMODE is disabled, it is recommended that this field be zero padded.
0x05	1	OPCODE	3 = Write to External EEPROM 5 = Write to External EEPROM 6 = Write to RAM location 8 = Write to serial port 11 = I ² C Write (v1.1) 13 = SPI Write (v1.1)
0x06	4	ADDR	Address argument for the above operations. Addresses are in 'big-endian' format, e.g., MSB comes first.
0x0A	1	WRITEDATA	Data to be written to the above address—must be from 1-30 bytes in length for UDP packets and 1-20 bytes for TCP packets

Table 5-3: ICP Packet – Write Request

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	1	STATUS	Will be equal to 0xFF if successful. Future release will contain additional error codes.
0x01	1	SEQNUM	Serial Sequence number. This is only present in response to op-codes 8 or 9. It contains the sequence number of the serial command just sent.

Table 5-4: ICP Packet – Write Response

Byte #	Length (Bytes)	Field Name	Description/Valid Values
0x00	5	PASSWD	Five character ASCII password—required if secure mode (SECMODE) is enabled.
0x05	1	OPCODE	14 = Reset device (soft reboot)
0x06	4	ADDR	These bytes are ignored for the RESET command

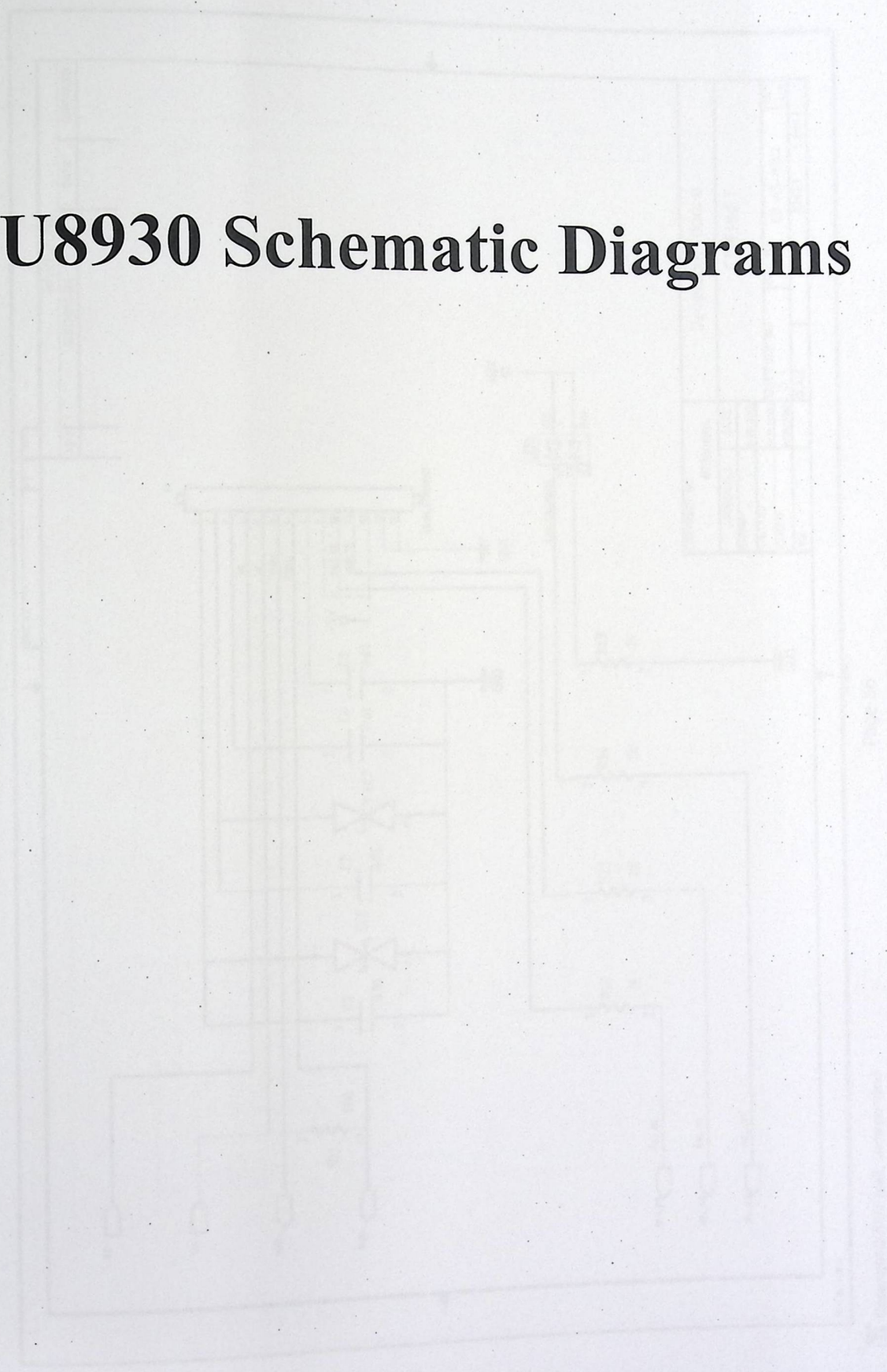
Table 5-5: Reset Device Request

Serial OPCODE List	
2	= Read from External EEPROM
3	= Write to External EEPROM
4	= Read from Internal EEPROM
5	= Write to Internal EEPROM
6	= Write to RAM location
7	= Read from RAM location
8	= Write to serial port
9	= Read from serial port
10	= I ² C Read (v1.1)
11	= I ² C Write (v1.1)
12	= SPI Read (v1.1)
13	= SPI Write (v1.1)
14	= Reset device (soft reboot)

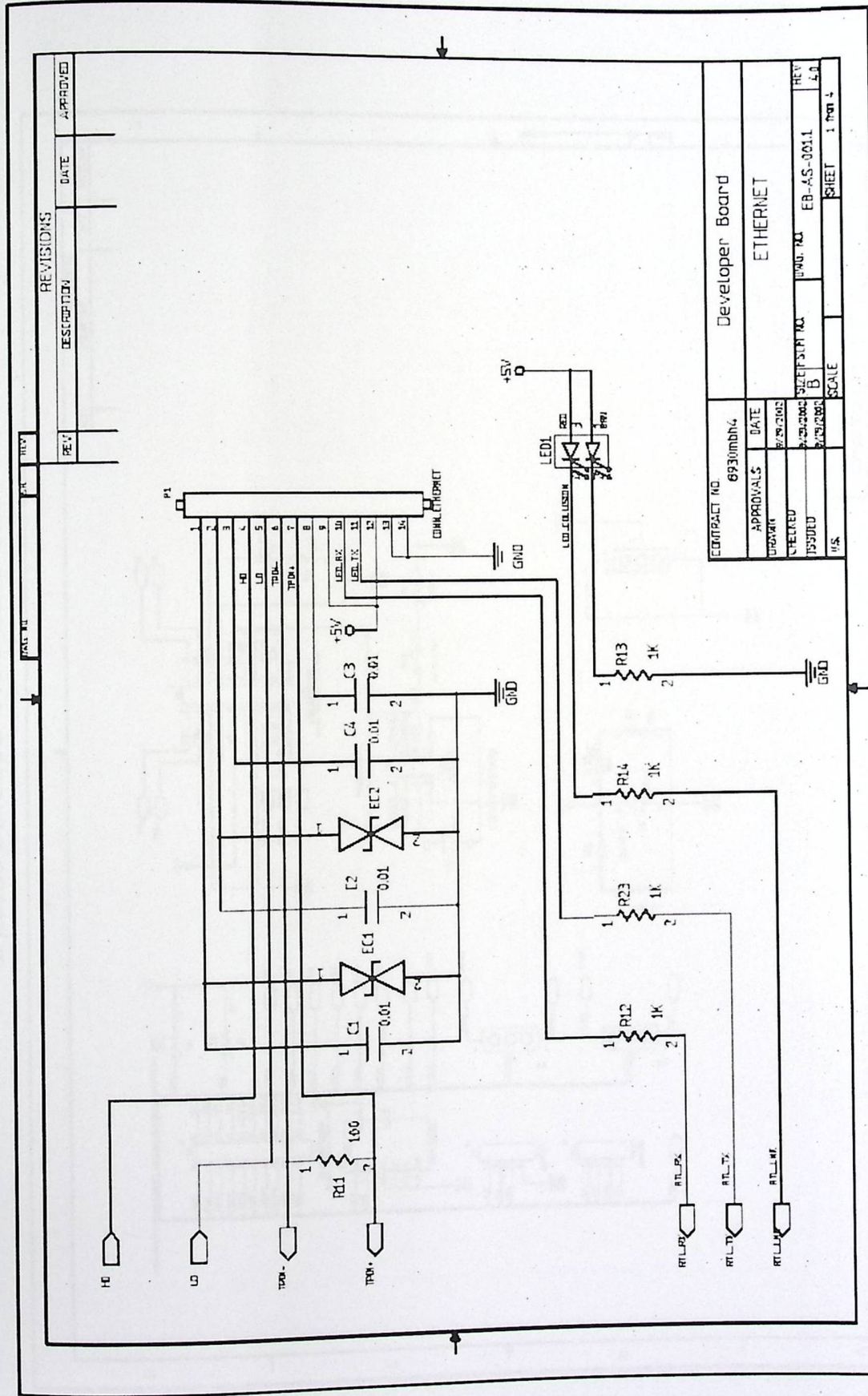
Table 5-6: Complete ICP OPCODE List

IPU8930 Schematic Diagrams

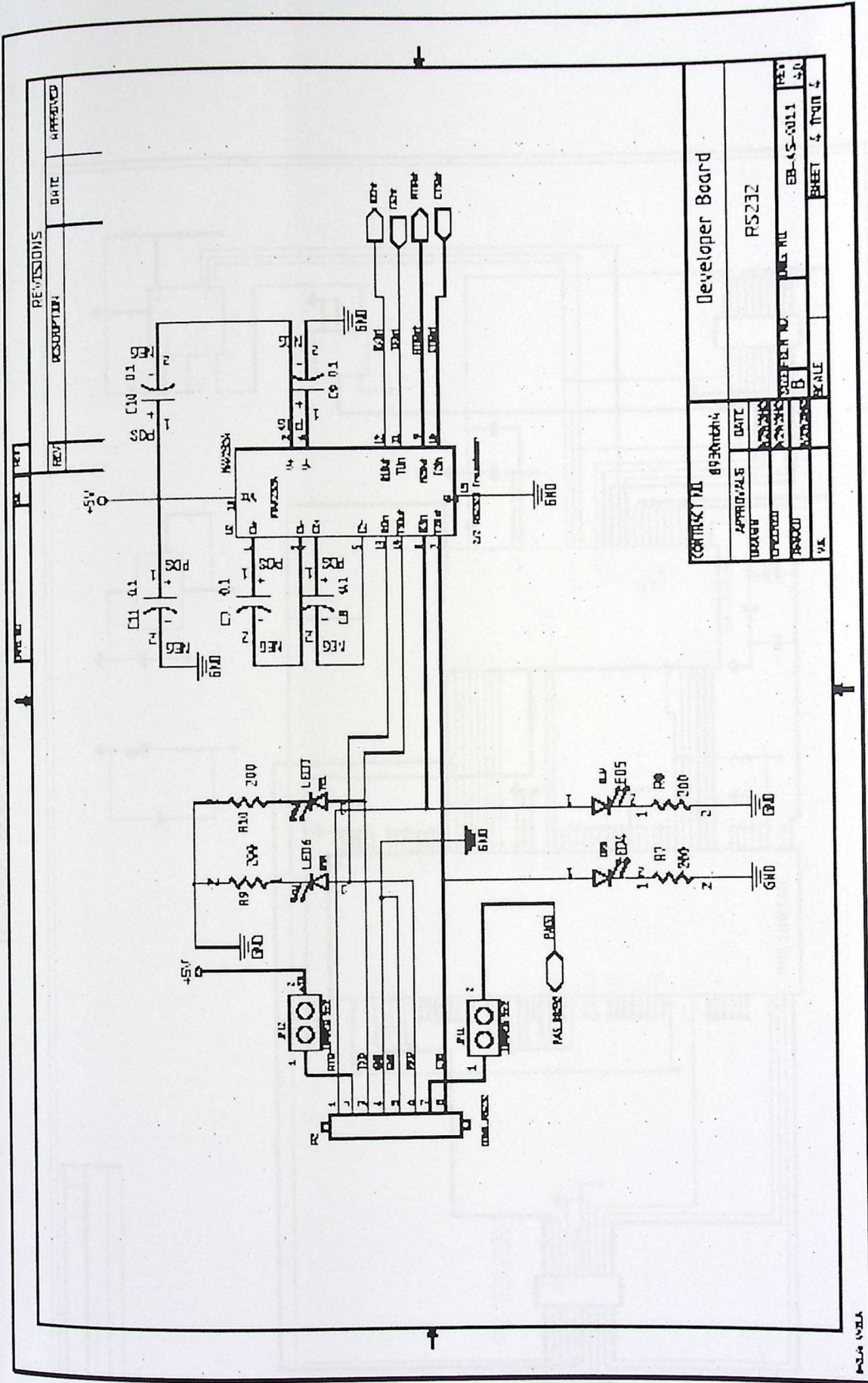
IPU8930 Developer Board Schematics



IPu8930 Developer Board Schematics



IPu8930 Developer Board Schematic



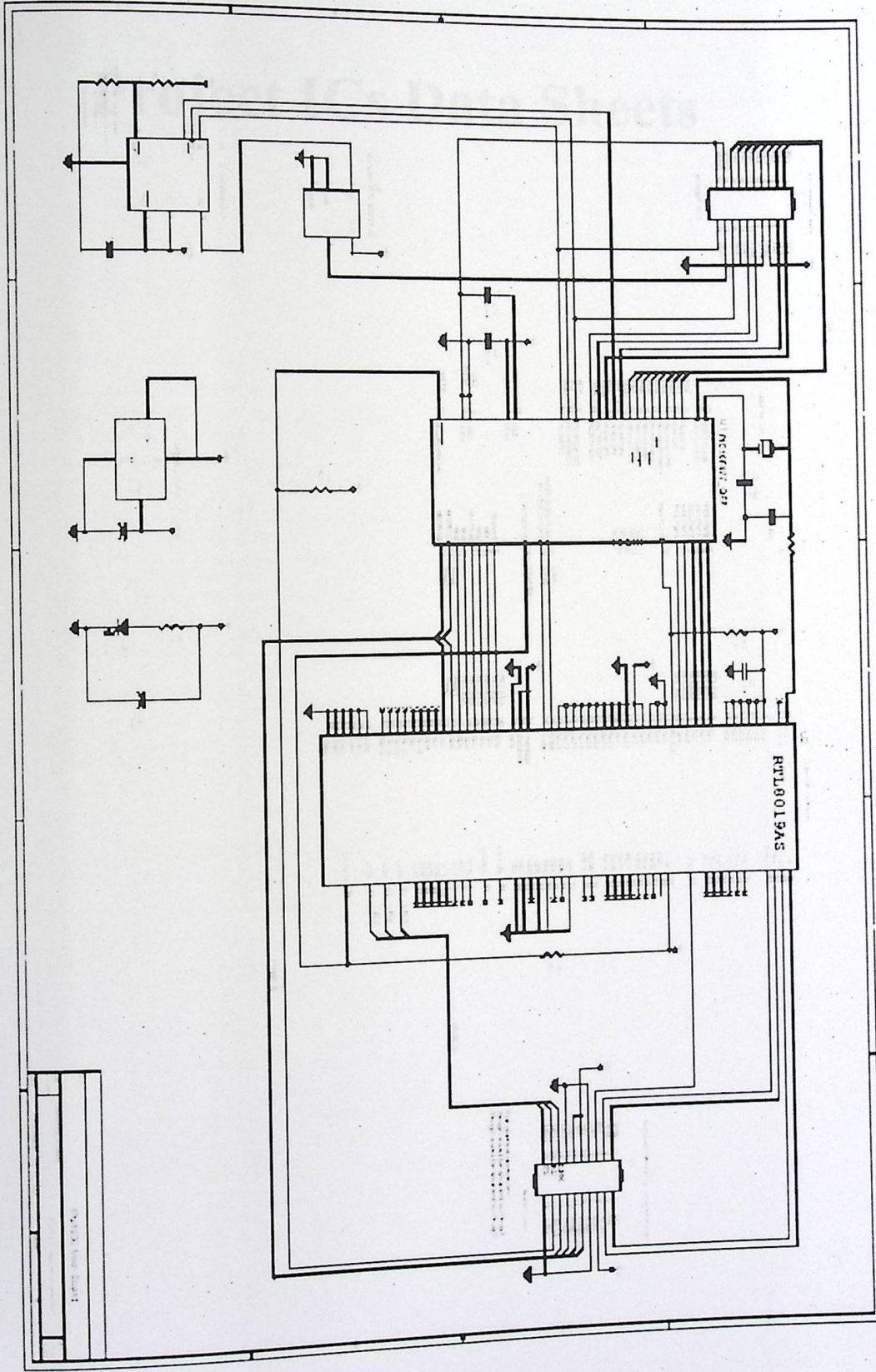
REVISIONS			
REV	DESCRIPTION	DATE	APPROVED

CONTROLLER		8930	
APPROVALS	DATE		
DESIGNER			
CHECKER			
APPROVER			
SCALE			
PROJECT NO.		EB-45-0011	
SHEET		4 OF 4	

Developer Board

RS232

IP_μ8930 Board Schematic



Project ICs Data Sheets

SDLS058

**SN54157, SN54LS157, SN54LS158, SN54S157, SN54S158,
SN74157, SN74LS157, SN74LS158, SN74S157, SN74S158**
QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

MARCH 1974 - REVISED MARCH 1988

- Buffered Inputs and Outputs
- Three Speed/Power Ranges Available

TYPES	TYPICAL AVERAGE PROPAGATION TIME	TYPICAL POWER DISSIPATION
'157	9 ns	150 mW
LS157	9 ns	49 mW
'S157	5 ns	250 mW
'LS158	7 ns	24 mW
'S158	4 ns	135 mW

applications

- Expand Any Data Input Point
- Multiplex Dual Data Buses
- Generate Four Functions of Two Variables (One Variable Is Common)
- Source Programmable Counters

description

These monolithic data selectors/multiplexers contain inverters and drivers to supply full on-chip data selection to the four output gates. A separate strobe input is provided. A 4 bit word is selected from one of two sources and is routed to the four outputs. The '157, 'LS157, and 'S157 present true data whereas the LS158 and 'S158 present inverted data to minimize propagation delay time.

FUNCTION TABLE

STROBE S	SELECT A/B	INPUTS		OUTPUT Y	
		A	B	'157, 'LS157, 'S157	'LS158 'S158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

H = High level, L = Low level, X = irrelevant

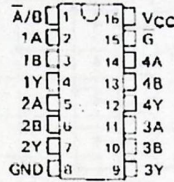
absolute maximum ratings over operating free air temperature range (unless otherwise noted)

Supply voltage, V _{CC} (See Note 1)	7 V
Input voltage: '157, 'S158	5.5 V
LS157, 'LS158	7 V
Operating free-air temperature range: SN54'	-55°C to 125°C
SN74'	0°C to 70°C
Storage temperature range	-65°C to 150°C

NOTE 1. Voltage values are with respect to network ground terminal.

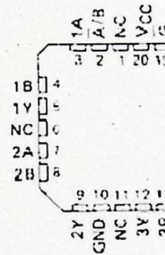
SN54157, SN54LS157, SN54S157,
SN54LS158, SN54S158 ... J OR W PACKAGE
SN74157 ... N PACKAGE
SN74LS157, SN74S157,
SN74LS158, SN74S158 ... D OR N PACKAGE

(TOP VIEW)



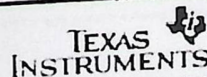
SN54LS157, SN54S157, SN54LS158,
SN54S158 ... FK PACKAGE

(TOP VIEW)



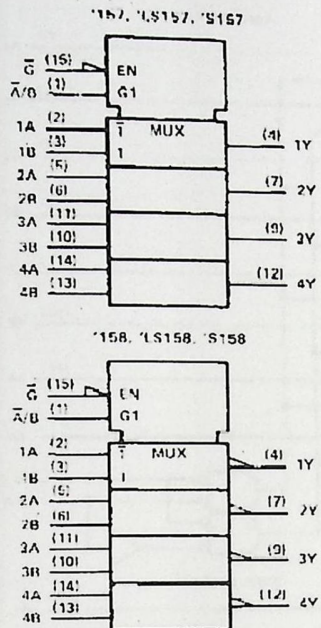
NC - No internal connection

PRODUCTION DATA documents contain information current as of publication date. Products conform to specific orders and the terms of Texas Instruments standard warranty. Production processing does not include testing of all parameters.

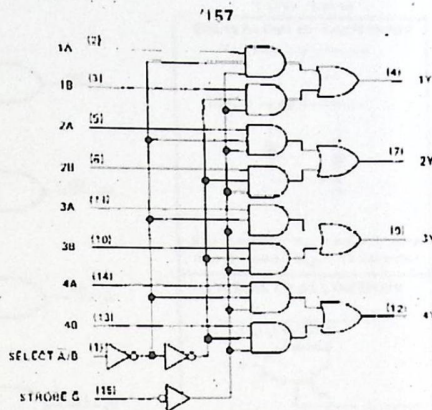


**SN54157, SN54LS157, SN54LS158, SN54S157, SN54S158,
SN74157, SN74LS157, SN74LS158, SN74S157, SN74S158
QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS**

logic symbols†

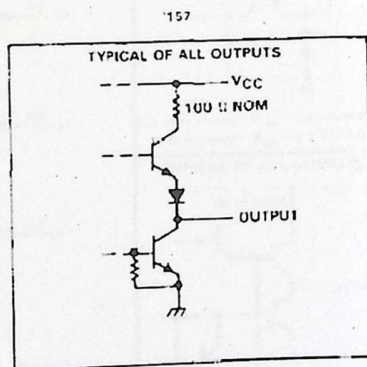
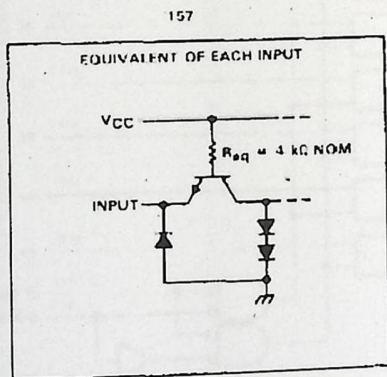


logic diagram (positive logic)



† These symbols are in accordance with ANSI/IEEE Std. 91-1984 and IEC Publication G17.12.
Pin numbers shown are for D, J, N, and W packages.

schematics of inputs and outputs

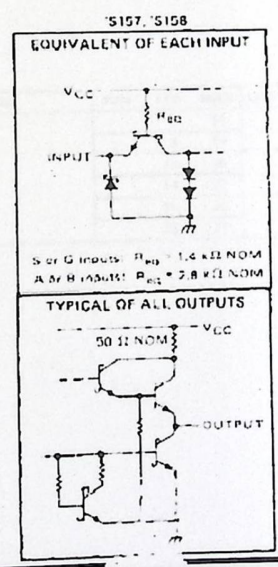
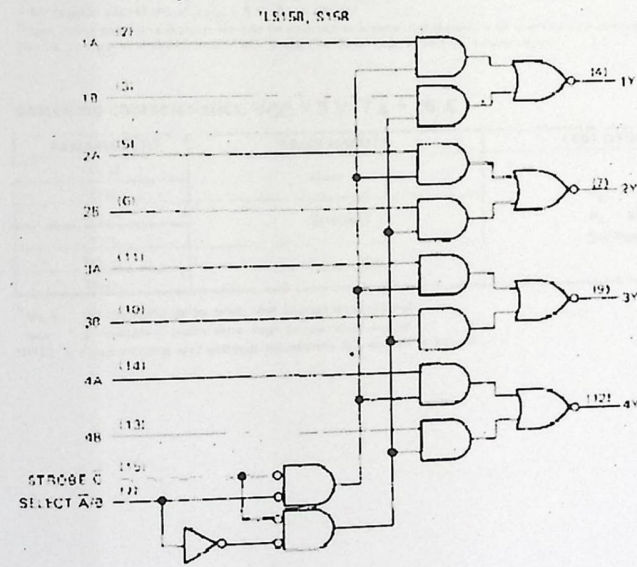
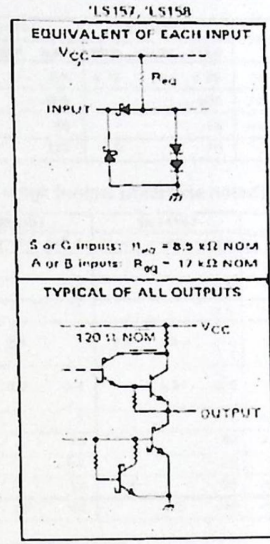
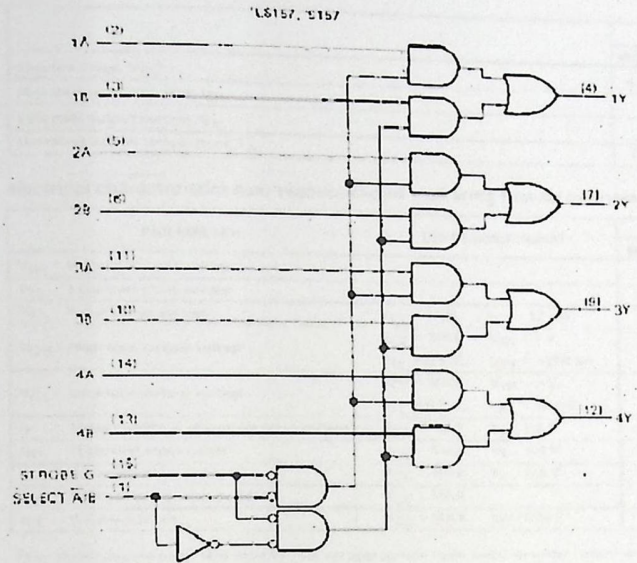


TEXAS INSTRUMENTS
POST OFFICE BOX 650312 • DALLAS, TEXAS 75265

SN54LS157, SN54LS158, SN54S157, SN54S158,
SN74LS157, SN74LS158, SN74S157, SN74S158
QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

logic diagrams (positive logic)

schematics of inputs and outputs



Pin numbers shown are for D, J, N, and W packages

TEXAS
INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

SN54157, SN74157
QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

recommended operating conditions

	SN54157			SN74157			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High level output current, I_{OH}			-800			-800	μ A
Low level output current, I_{OL}			16			16	mA
Operating free air temperature, T_A	-55	125		0	70		$^{\circ}$ C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS ¹	SN54157			SN74157			UNIT
		MIN	TYP ²	MAX	MIN	TYP ²	MAX	
V_{IH} High level input voltage		2			2			V
V_{IL} Low level input voltage				0.8			0.8	V
V_{IN} Input clamp voltage	$V_{CC} = \text{MIN.}$, $I_i = 12 \text{ mA}$			1.5			1.5	V
V_{OH} High level output voltage	$V_{CC} = \text{MIN.}$, $V_{IH} = 2 \text{ V.}$ $V_{IL} = 0.8 \text{ V.}$, $I_{OH} = -800 \mu\text{A}$	2.4	3.4		2.4	3.4		V
V_{OL} Low level output voltage	$V_{CC} = \text{MIN.}$, $V_{IH} = 2 \text{ V.}$ $V_{IL} = 0.8 \text{ V.}$, $I_{OL} = 16 \text{ mA}$		0.2	0.4		0.2	0.4	V
I_i Input current at maximum input voltage	$V_{CC} = \text{MAX.}$, $V_i = 5.5 \text{ V}$			1			1	mA
I_{IH} High level input current	$V_{CC} = \text{MAX.}$, $V_i = 2.4 \text{ V}$			40			40	μ A
I_{IL} Low level input current	$V_{CC} = \text{MAX.}$, $V_i = 0.4 \text{ V}$			-16			-16	μ A
I_{OS} Short circuit output current ³	$V_{CC} = \text{MAX}$	-20		55	-18		55	mA
I_{CC} Supply current	$V_{CC} = \text{MAX}$ See Note 2		30	48		30	48	mA

¹ For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

² All typical values are at $V_{CC} = 5 \text{ V.}$, $T_A = 25^{\circ}\text{C.}$

³ Not more than one output should be shorted at a time and duration of short-circuit should not exceed one second.

NOTE 2: I_{CC} is measured with 4.5 V applied to all inputs and all outputs open.

switching characteristics, $V_{CC} = 5 \text{ V.}$ $T_A = 25^{\circ}\text{C}$

PARAMETER ¹	FROM (INPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Data	$C_L = 15 \text{ pF.}$ $R_L = 400 \Omega.$ See Note 3		9	14	ns
t_{PHL}			13	20		
t_{SPLH}	Strobe \bar{G}			14	21	ns
t_{SPLH}			15	23		
t_{SPLH}	Select \bar{A}/\bar{B}			18	27	ns
t_{SPLH}						

¹ t_{PLH} - propagation delay time, low to high level output

² t_{PHL} - propagation delay time, high to low level output

NOTE 3: Load circuits and voltage waveforms are shown in Section 1

SN54LS157, SN54LS158, SN74LS157, SN74LS158
QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

recommended operating conditions

PARAMETER	SN54LS*			SN74LS*			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
V _{CC} Supply voltage	4.5	5	5.5	4.75	5	5.25	V
I _{OH} High-level output current			-400			-400	μA
I _{OL} Low-level output current			4			8	mA
T _A Operating free-air temperature	-55	125	0			70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS†	SN54LS*			SN74LS*			UNIT
		MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V _{IH} High-level input voltage		2			2			V
V _{IL} Low-level input voltage				0.7			0.9	V
V _{IK} Input clamp voltage	V _{CC} = MIN, I _I = -18 mA			-1.5			-1.5	V
V _{OH} High-level output voltage	V _{CC} = MIN, V _{IH} = 2 V, V _{IL} = MAX, I _{OH} = -400 μA	2.5	3.4		2.7	3.4		V
V _{OL} Low-level output voltage	V _{CC} = MIN, V _{IH} = 2 V, I _{OL} = 4 mA, I _{OL} = 8 mA		0.25	0.4		0.25	0.4	V
I _I Input current at maximum input voltage	A/B or \bar{G}				0.2		0.2	mA
	A or B	V _{CC} = MAX, V _I = 1 V			0.1		0.1	
I _{IH} High-level input current	A/B or \bar{G}		40		40			μA
	A or B	V _{CC} = MAX, V _I = 2.7 V		20		20		
I _{IL} Low-level input current	A/B or \bar{G}		-0.9		-0.9			mA
	A or B	V _{CC} = MAX, V _I = 0.4 V		-0.4		-0.4		
I _{OS} Short-circuit output current‡	V _{CC} = MAX	-20		100	-20		100	mA
I _{CC} Supply current	V _{CC} = MAX, See Note 2	LS157	9.7	16	9.7	16		mA
		LS158	4.8	8	4.8	8		
	V _{CC} = MAX, All A inputs at 4.5 V, All other inputs at 0 V	LS158	6.5	11	6.5	11		

* For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.

† As typical values are at V_{CC} = 5 V, T_A = 25°C.

‡ Not more than one output should be shorted at a time and duration of short circuit should not exceed one second.

NOTE 2: I_{CC} is measured with 4.5 V applied to all inputs and all outputs open.

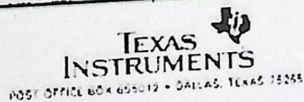
switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER	FROM (INPUT)	TEST CONDITIONS	LS157			LS158			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t _{PLH}	Data	C _L = 15 pF, R _L = 2 kΩ, See Note 1	8	14		7	12		ns
t _{PHL}			9	14		10	15		
t _{PLH}	Strobe \bar{G}		13	20		11	17		ns
t _{PHL}			14	21		18	24		
t _{PLH}	Select A, B		15	23		13	20		ns
t _{PHL}			18	27		16	24		

t_{PLH} = propagation delay time, low to high-level output

t_{PHL} = propagation delay time, high-to-low-level output

NOTE 3: Load circuits and voltage diagrams are shown in Section 1.



SN54S157, SN54S158, SN74S157, SN74S158
QUADRUPLE 2-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

recommended operating conditions

	SN54S157 SN54S158			SN74S157 SN74S158			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			-1			-1	mA
Low-level output current, I_{OL}			20			20	mA
Operating free-air temperature, T_A	-55	125		0	70		$^{\circ}$ C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS ¹	SN54S157 SN74S157			SN54S158 SN74S158			UNIT
		MIN	TYP ²	MAX	MIN	TYP ²	MAX	
V_{IH} High-level input voltage		2			2			V
V_{IL} Low-level input voltage				0.8			0.8	V
V_{IK} Input clamp voltage	$V_{CC} - \text{MIN}$, $I_I = -18 \text{ mA}$			-1.2			-1.2	V
V_{OH} High-level output voltage	$V_{CC} - \text{MIN}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = 0.8 \text{ V}$, $I_{OH} = -1 \text{ mA}$	Series 54S 2.5	Series 54S 3.4		Series 74S 2.5	Series 74S 3.4		V
V_{OL} Low-level output voltage	$V_{CC} - \text{MIN}$, $V_{IH} = 2 \text{ V}$, $V_{IL} = 0.8 \text{ V}$, $I_{OL} = 20 \text{ mA}$			0.5			0.5	V
I_I Input current at maximum input voltage	$V_{CC} - \text{MAX}$, $V_I = 5.5 \text{ V}$			1			1	mA
I_{IH} High-level input current	\bar{A}/\bar{B} or \bar{B} $V_{CC} - \text{MAX}$, $V_I = 2.7 \text{ V}$			100			100	μ A
I_{IL} Low-level input current	\bar{A}/\bar{B} or \bar{G} $V_{CC} - \text{MAX}$, $V_I = 0.5 \text{ V}$			50			50	μ A
I_{IS} Short-circuit output current	$V_{CC} - \text{MAX}$ $V_{CC} - \text{MAX}$, All inputs at 4.5 V, See Note 2			40	100	40	100	mA
I_{CC} Supply current	$V_{CC} - \text{MAX}$, A inputs at 4.5 V, B,C,S inputs at 0 V, See Note 2			50	78	39	61	mA

¹ For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
² All typical values are at $V_{CC} = 5 \text{ V}$, $T_A = 25^{\circ}\text{C}$.
³ Not more than one output should be shorted at a time, and duration of the short circuit should not exceed one second.
⁴ Note 2: I_{CC} is measured with all outputs open.

switching characteristics, $V_{CC} = 5 \text{ V}$, $T_A = 25^{\circ}\text{C}$

PARAMETER ¹	FROM (INPUT)	TEST CONDITIONS	SN54S157 SN74S157			SN54S158 SN74S158			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	Data	$C_L = 15 \text{ pF}$, $R_L = 280 \Omega$, See Note 3	5	7.5		4	6		ns
t_{PHL}	Data		4.5	6.5		4	6		ns
t_{PLH}	Strobe \bar{G}		8.5	12.5		6.5	11.5		ns
t_{PHL}	Strobe \bar{G}		7.5	12		7	12		ns
t_{PLH}	Select A/B		4.4	15		8	12		ns
t_{PHL}	Select A/B		9.5	15		8	12		ns

¹ t_{PLH} = propagation delay time, low-to-high-level output
² t_{PHL} = propagation delay time, high-to-low-level output
NOTE 3: Load circuits and voltage waveforms are shown in Section 1.

TEXAS
INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

DM74LS90 Decade and Binary Counters

General Description

Each of these monolithic counters contains four master-slave flip-flops and additional gating to provide a divide-by-two counter and a three-stage binary counter for which the count cycle length is divide-by-five for the DM74LS90.

All of these counters have a gated zero reset and the DM74LS90 also has gated set-to-nine inputs for use in BCD nine's complement applications.

To use their maximum count length (decade or four bit binary), the B input is connected to the Q_A output. The input count pulses are applied to input A and the outputs are as described in the appropriate truth table. A symmetrical divide-by-ten count can be obtained from the DM74LS90 counters by connecting the Q_D output to the A input and applying the input count to the B input which gives a divide-by-ten square wave at output Q_A .

Features

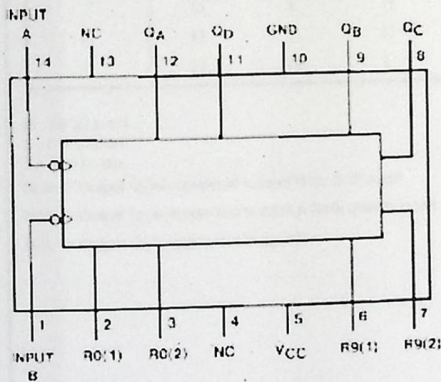
- Typical power dissipation 45 mW
- Count frequency 42 MHz

Ordering Code:

Order Number	Package Number	Package Description
DM74LS90M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
DM74LS90N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagram



Reset/Count Truth Table

Reset Inputs				Output			
R0(1)	R0(2)	R9(1)	R9(2)	Q_D	Q_C	Q_B	Q_A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

Function Tables

BCD Count Sequence (Note 1)

Count	Output			
	Q _D	Q _C	Q _B	Q _A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

Bi-Quinary (5-2) (Note 2)

Count	Output			
	Q _A	Q _D	Q _C	Q _B
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

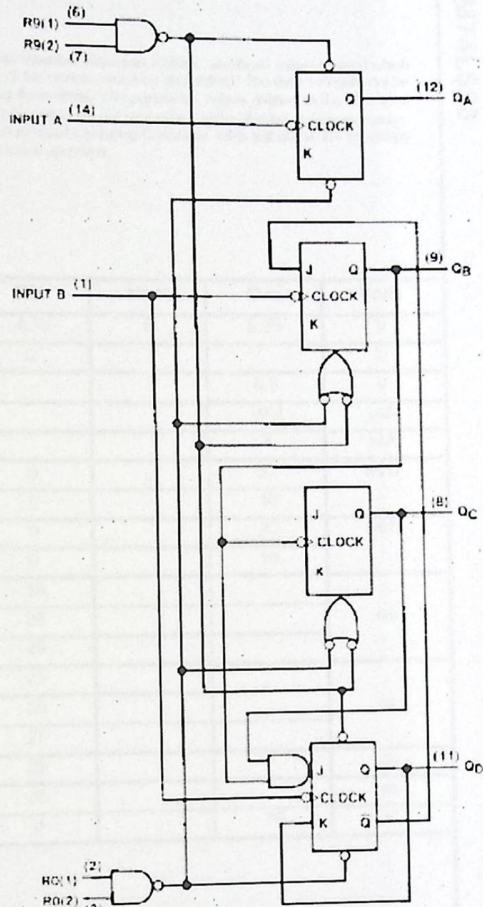
H = HIGH Level
 L = LOW Level
 X = Don't Care

Note 1: Output Q_D is connected to input B for BCD count.

Note 2: Output Q_C is connected to input A for bi-quinary count.

Note 3: Output Q_C is connected to input B

Logic Diagram



The J and K inputs shown without connection are for reference only and are functionally at a high level.

Absolute Maximum Ratings (Note 4)

Supply Voltage	7V
Input Voltage (Reset)	7V
Input Voltage (A or B)	5.5V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	65°C to -150°C

Note 4: The 'Absolute Maximum Ratings' are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the 'Electrical Characteristics' table are not guaranteed at the absolute maximum ratings. The 'Recommended Operating Conditions' table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			-0.4	mA
I_{OL}	LOW Level Output Current			8	mA
f_{CLK}	Clock Frequency (Note 5)	A to Q_A	0	32	MHz
		B to Q_B	0	16	
f_{CLK}	Clock Frequency (Note 6)	A to Q_A	0	20	MHz
		B to Q_B	0	10	
t_W	Pulse Width (Note 5)	A	15		ns
		B	30		
		Reset	15		
t_W	Pulse Width (Note 6)	A	25		ns
		B	50		
		Reset	25		
t_{RH}	Reset Release Time (Note 5)	25			ns
t_{RH}	Reset Release Time (Note 6)	35			ns
T_A	Free Air Operating Temperature	0		70	°C

Note 5: $C_i = 15$ pF, $R_i = 2$ k Ω , $T_A = 25$ °C and $V_{CC} = 5$ V.

Note 6: $C_i = 50$ pF, $R_i = 2$ k Ω , $T_A = 25$ °C and $V_{CC} = 5$ V.

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 7)	Max	Units
V_i	Input Clamp Voltage	$V_{CC} = \text{Min}$, $I_i = -18$ mA			-1.5	V
V_{OH}	HIGH Level Output Voltage	$V_{CC} = \text{Min}$, $I_{OH} = \text{Max}$ $V_{IL} = \text{Max}$, $V_{IH} = \text{Min}$	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	$V_{CC} = \text{Min}$, $I_{OL} = \text{Max}$ (Note 8) $V_{IL} = \text{Max}$, $V_{IH} = \text{Min}$ $I_{OL} = 4$ mA, $V_{CC} = \text{Min}$		0.35	0.5	V
					0.25	
I_i	Input Current @ Max Input Voltage	$V_{CC} = \text{Max}$, $V_i = 7$ V	Reset		0.1	mA
		$V_{CC} = \text{Max}$ $V_i = 5.5$ V	A		0.2	
			B		0.4	
I_{IH}	HIGH Level Input Current	$V_{CC} = \text{Max}$, $V_i = 2.7$ V	Reset		20	μ A
			A		40	
			B		80	
I_{IL}	LOW Level Input Current	$V_{CC} = \text{Max}$, $V_i = 0.4$ V	Reset		-0.4	mA
			A		-2.4	
			B		-3.2	
I_{SC}	Short Circuit Output Current	$V_{CC} = \text{Max}$ (Note 9)	-20	9	15	mA
I_{CC}	Supply Current	$V_{CC} = \text{Max}$ (Note 7)				mA

Note 7: All t_{RH} values are at $V_{CC} = 5$ V, $T_A = 25$ °C.

Electrical Characteristics (Continued)

Note 8: Q_A outputs are tested at $L_{CL} = t_{max}$ plus the limit value of t_{CL} for the B input. This permits driving the B input while maintaining full fan out capability.

Note 9: Not more than one output should be checked at a time, and the duration should not exceed one second.

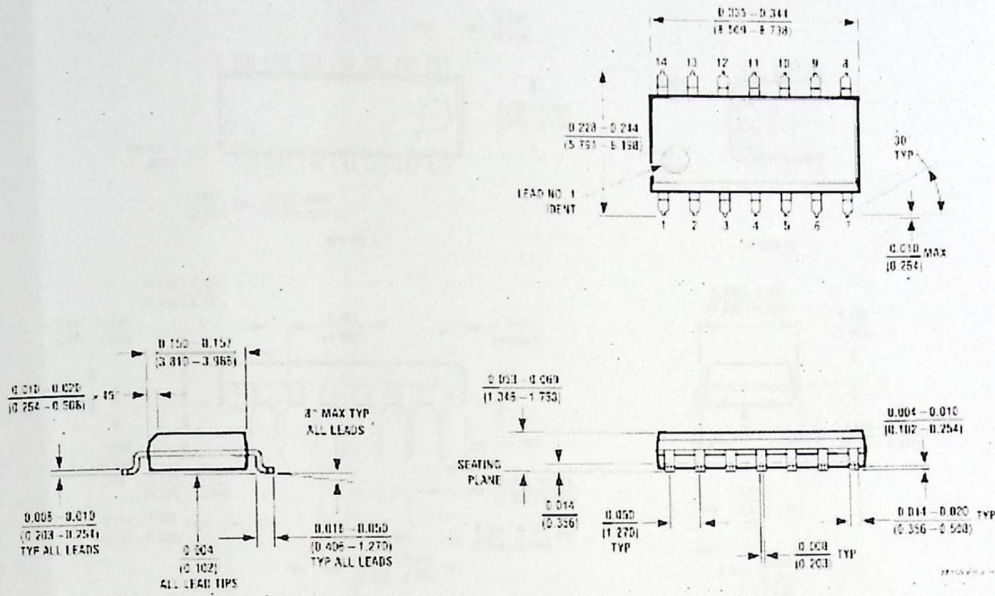
Note 10: t_{CL} is measured with all outputs open, both RO inputs grounded following momentary connection to 4.5V and all other inputs grounded.

Switching Characteristics at $V_{CC} = 5V$ and $T_A = 25^\circ C$

Symbol	Parameter	From (Input) To (Output)	$R_L = 2 k\Omega$				Units
			$C_L = 15 pF$		$C_L = 50 pF$		
			Min	Max	Min	Max	
f_{max}	Maximum Clock Frequency	A to Q_A B to Q_B	32 16		20 10		MHz
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	A to Q_A		16		20	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	A to Q_A		18		24	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	A to Q_B		48		52	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	A to Q_B		50		60	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	B to Q_B		16		23	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	B to Q_B		21		30	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	B to Q_C		32		37	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	B to Q_C		35		44	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	B to Q_D		32		36	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	B to Q_D		35		44	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	SET-0 to Q_A, Q_B		30		35	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	SET-0 to Q_B, Q_C		40		48	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	SET-0 to Any Q		40		52	ns

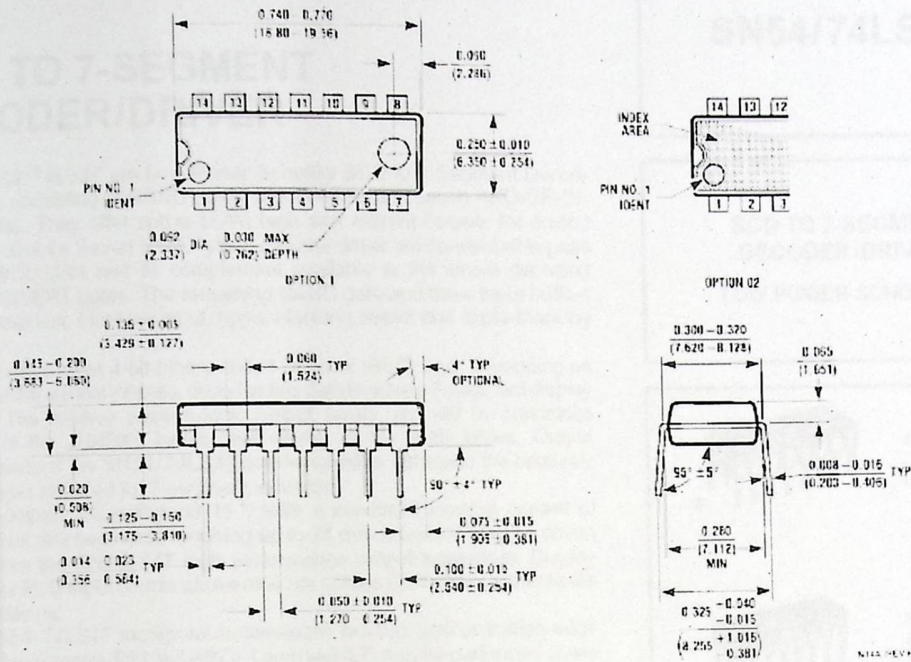
Physical Dimensions inches (millimeters) unless otherwise noted

DM74LS90



14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-120, 0.150 Narrow
Package Number M14A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide Package Number N14A

Fairchild does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and Fairchild reserves the right at any time without notice to change said circuitry and specifications.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

www.fairchildsemi.com



BCD TO 7-SEGMENT DECODER/DRIVER

The SN54/74LS47 are Low Power Schottky BCD to 7-Segment Decoder/Drivers consisting of NAND gates, input buffers and seven AND-OR-INVERT gates. They offer active LOW, high sink current outputs for driving indicators directly. Seven NAND gates and one driver are connected in pairs to make BCD data and its complement available to the seven decoding AND-OR-INVERT gates. The remaining NAND gate and three input buffers provide lamp test, blanking input, ripple-blanking output and ripple-blanking input.

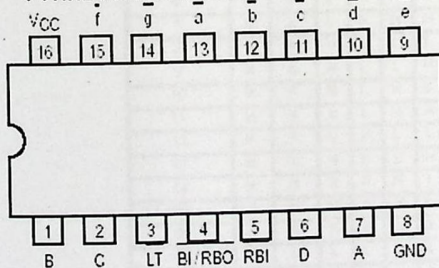
The circuits accept 4-bit binary-coded-decimal (BCD) and, depending on the state of the auxiliary inputs, decodes this data to drive a 7-segment display indicator. The relative positive-logic output levels, as well as conditions required at the auxiliary inputs, are shown in the truth tables. Output configurations of the SN54/74LS47 are designed to withstand the relatively high voltages required for 7-segment indicators.

These outputs will withstand 15 V with a maximum reverse current of 250 μ A. Indicator segments requiring up to 24 mA of current may be driven directly from the SN74LS47 high performance output transistors. Display patterns for BCD input counts above nine are unique symbols to authenticate input conditions.

The SN54/74LS47 incorporates automatic leading and/or trailing-edge zero-blanking control (RBI and RBO). Lamp test (LT) may be performed at any time which the BI/RBO node is a HIGH level. This device also contains an overriding blanking input (BI) which can be used to control the lamp intensity by varying the frequency and duty cycle of the BI input signal or to inhibit the outputs.

- Lamp Intensity Modulation Capability (BI/RBO)
- Open Collector Outputs
- Lamp Test Provision
- Leading/Trailing Zero Suppression
- Input Clamp Diodes Limit High-Speed Termination Effects

CONNECTION DIAGRAM DIP (TOP VIEW)



PIN NAMES

A, B, C, D	BCD Inputs
RBI	Ripple-Blanking Input
LT	Lamp-Test Input
BI/RBO	Blanking Input or Ripple-Blanking Output
a, b, c, d, e, f, g	Outputs

LOADING (Note a)

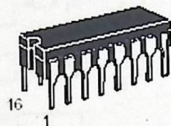
	HIGH	LOW
A, B, C, D	0.5 U.L.	0.25 U.L.
RBI	0.5 U.L.	0.25 U.L.
LT	0.5 U.L.	0.25 U.L.
BI/RBO	0.5 U.L.	0.75 U.L.
a, b, c, d, e, f, g	1.2 U.L.	2.0 U.L.
Outputs	Open-Collector	15 (7.5) U.L.

NOTES:

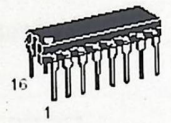
- a) 1 Unit Load (U.L.) = 40 μ A HIGH, 1.6 mA LOW.
 b) Output current measured at $V_{OUT} = 0.5$ V.
 The Output LOW drive factor is 7.5 U.L. for Military (54) and 15 U.L. for Commercial (74) Temperature Ranges.

SN54/74LS47

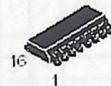
BCD TO 7-SEGMENT DECODER/DRIVER LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09



N SUFFIX
PLASTIC
CASE 648-08

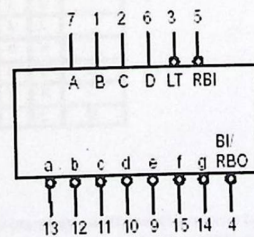


D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

LOGIC SYMBOL



V_{CC} = PIN 16
GND = PIN 8

FAST AND LS TTL DATA



BCD TO 7-SEGMENT DECODER DRIVER

The 7447 is a BCD-to-7-segment decoder driver. It is designed to drive common cathode 7-segment displays. The device has four BCD inputs (A, B, C, D) and seven outputs (a, b, c, d, e, f, g) corresponding to the segments of the display. The outputs are active-low, meaning they sink current from the display segments.



FIGURE 1. PIN DESIGNATIONS — RESULTANT DISPLAYS

TRUTH TABLE

INPUTS	OUTPUTS							NOTE						
	A	B	C	D	a	b	c		d	e	f	g		
0	H	H	L	L	L	L	H	L	L	L	L	L	H	A
1	H	X	L	L	L	H	H	L	L	H	H	H	H	A
2	H	X	L	L	H	L	H	L	L	L	L	L	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	
4	H	X	L	H	L	L	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	L	L	
6	H	X	L	H	H	L	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	
8	H	X	H	L	L	L	H	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	
10	H	X	H	L	H	L	H	L	L	L	H	L	L	
11	H	X	H	L	H	H	H	L	L	L	H	H	L	
12	H	X	H	H	L	L	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	L	L	
14	H	X	H	H	H	L	H	H	H	L	L	L	L	
15	X	X	X	X	X	X	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	B
PE	H	L	L	L	L	L	L	H	H	H	H	H	H	C
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	D

H = HIGH Voltage Level
L = LOW Voltage Level
X = Indifferent

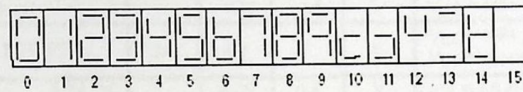
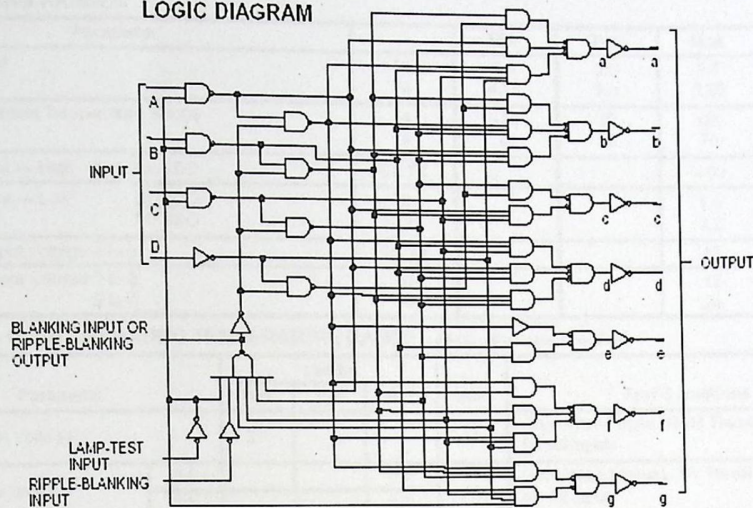
NOTES:

- BI (BIAS) is an active-low logic input. When BI is held at a HIGH level, all segment outputs are active-low. When BI is held at a LOW level, all segment outputs are active-high. When BI is held at a HIGH level, the blanking output (BI) is active-low. When BI is held at a LOW level, the blanking output (BI) is active-high.
- When a LOW level is applied to the blanking input (BI), all segment outputs go to a LOW level regardless of the state of any other input condition.
- When the blanking input (BI) and inputs A, B, C, and D are all LOW level, with the lamp test input (LT) at a HIGH level, all segment outputs go to a HIGH level and the blanking output (BI) goes to a LOW level (response code 0).
- When the blanking input (BI) is active-low, all segment outputs go to a LOW level, and all segment outputs go to a HIGH level when the blanking input (BI) is active-high.

FAST AND LS TTL DATA

SN54/74LS47

LOGIC DIAGRAM



NUMERICAL DESIGNATIONS — RESULTANT DISPLAYS

TRUTH TABLE

DECIMAL OR FUNCTION	INPUTS						OUTPUTS							NOTE	
	LT	RBI	D	C	B	A	BIRBO	a	b	c	d	e	f		g
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H	A
1	H	X	L	L	L	H	H	H	L	L	H	L	H	H	A
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L	
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L	
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L	
5	H	X	L	H	L	H	H	L	H	L	L	L	H	L	
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L	
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H	
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L	
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L	
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L	
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L	
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L	
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L	
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L	
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H	
BI	X	X	X	X	X	X	L	H	H	H	H	H	H	H	B
RBI	H	L	L	L	L	L	L	H	H	H	H	H	H	H	C
LT	L	X	X	X	X	X	H	L	L	L	L	L	L	L	D

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

NOTES:

- BI/RBO is wire-AND logic serving as Blanking Input (BI) and/or ripple-blanking output (RBO). The blanking out (BI) must be open or held at a HIGH level when output functions 0 through 15 are desired, and ripple-blanking input (RBI) must be open or at a HIGH level if blanking of a decimal 0 is not desired. X = input may be HIGH or LOW.
- When a LOW level is applied to the blanking input (forced condition) all segment outputs go to a LOW level regardless of the state of any other input condition.
- When ripple-blanking input (RBI) and inputs A, B, C, and D are at LOW level, with the lamp test input at HIGH level, all segment outputs go to a HIGH level and the ripple-blanking output (RBO) goes to a LOW level (response condition).
- When the blanking input/ripple-blanking output (BI/RBO) is open or held at a HIGH level, and a LOW level is applied to lamp test input, all segment outputs go to a LOW level.

FAST AND LS TTL DATA

SN54/74LS47

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V _{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T _A	Operating Ambient Temperature Range	54 74	-55 0	25 25	125 70	°C
I _{OH}	Output Current — High BI/RBO	54, 74			-50	µA
I _{OL}	Output Current — Low BI/RBO BI/RBO	54 74			1.6 3.2	mA
V _{O(off)}	Off-State Output Voltage \bar{a} to \bar{g}	54, 74			15	V
I _{O(on)}	On-State Output Current \bar{a} to \bar{g} \bar{a} to \bar{g}	54 74			12 24	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
V _{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Threshold Voltage for All Inputs
V _{IL}	Input LOW Voltage	54		0.7	V	Guaranteed Input LOW Threshold Voltage for All Inputs
		74		0.8		
V _{IK}	Input Clamp Diode Voltage		-0.65	-1.5	V	V _{CC} = MIN, I _{IN} = -18 mA
V _{OH}	Output HIGH Voltage, BI/RBO	2.4	4.2		V	V _{CC} = MIN, I _{OH} = -50 µA, V _{IN} = V _{IH} or V _{IL} per Truth Table
V _{OL}	Output LOW Voltage BI/RBO	54, 74	0.25	0.4	V	I _{OL} = 1.6 mA V _{CC} = MIN, V _{IN} = V _{IH} or V _{IL} per Truth Table
		74	0.35	0.5		
I _{O(off)}	Off-State Output Current \bar{a} through \bar{g}			250	µA	V _{CC} = MAX, V _{IN} = V _{IH} or V _{IL} per Truth Table, V _{O(off)} = 15 V
V _{O(on)}	On-State Output Voltage \bar{a} through \bar{g}	54, 74	0.25	0.4	V	I _{O(on)} = 12 mA V _{CC} = MAX, V _{IN} = V _{IH} or V _{IL} per Truth Table
		74	0.35	0.5		
I _{IH}	Input HIGH Current			20	µA	V _{CC} = MAX, V _{IN} = 2.7 V
				0.1		
I _{IL}	Input LOW Current BI/RBO Any Input except BI/RBO			-1.2 -0.4	mA	V _{CC} = MAX, V _{IN} = 0.4 V
I _{OS, BI/RBO}	Output Short Circuit Current (Note 1)	-0.3		-2.0	mA	V _{CC} = MAX, V _{OUT} = 0 V
I _{CC}	Power Supply Current		7.0	13	mA	V _{CC} = MAX

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25 °C)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
t _{FHL}	Propagation Delay, Address Input to Segment Output			100	ns	V _{CC} = 5.0 V C _L = 15 pF
t _{PLH}	Propagation Delay, RBI Input to Segment Output			100	ns	
t _{FHL}	Propagation Delay, Address Input to Segment Output			100	ns	
t _{PLH}	Propagation Delay, RBI Input to Segment Output			100	ns	

AC WAVEFORMS

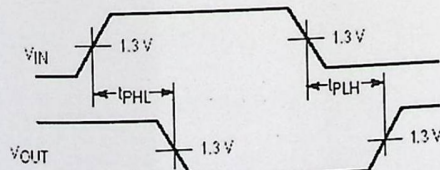


Figure 1

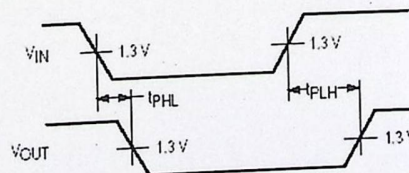


Figure 2

FAST AND LS TTL DATA

Project software source code Data Sheets

```
private int IP_ADDR = 192; // IP address part number

// Variables
int Address = 0; // null;
Socket sock = null; // Socket
InputStream in = null; // Input data stream
OutputStream out = null; // Output data stream instance

StringBuffer buffer = new StringBuffer();
Button read = new Button("Read");
Button select = new Button("Select");
Button on = new Button("Turn OFF");
Button off = new Button("Turn ON");
Label value = new Label();
Label cond = new Label();

// Arrays
// Write RAM variable ICP command
// Configure all channels as Digital Outputs
// Open the file = data, address is 0x00000000, bytes to write 1
byte[] setAllDigitalOutput =
```

```
import java.applet.Applet;
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
```

```
public class Start extends Applet
```

```
{
```

```
    // Constants
```

```
    private int IPSIL_PORT = 8930;           // Ipsil device port number
```

```
    // Variables
```

```
    InetAddress ia = null;
```

```
    Socket sock = null;           // Socket
```

```
    InputStream sIn = null;       // Input data stream
```

```
    OutputStream sOut = null;     // Output data stream instance
```

```
    StringBuffer buffer = new StringBuffer();
```

```
    Button read=new Button("Read");
```

```
    Button select=new Button("Select");
```

```
    Button on=new Button("Turn OFF");
```

```
    Button off=new Button("Turn ON");
```

```
    Label value=new Label();
```

```
    Label eCond=new Label();
```

```
    // Arrays
```

```
    // Write RAM variable ICP command
```

```
    // Configures all channels as Digital Outputs
```

```
    // Operation code = 0x06, address in RAM 0x113, bytes to write 2
```

```
    byte [] setAllDigitalOutput =
```

```

    { 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
      0x06, // "Write EEPROM" cmd (1 byte len)
      0x00, 0x00, 0x01, 0x13, // Start location (4 bytes len)
      (byte)0x86, // All
channels Digital (1 byte len)
      0x0f, // 1,2,3,4
input 4,5,6,7 output (1 byte len)
};
byte [] t =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
  0x06, // "Write EEPROM" cmd (1 byte len)
  0x00, 0x00, 0x01, 0x13, // Start location (4 bytes len)
  (byte)0x86, // All
channels Digital (1 byte len)
      0x00, // All
channels as Outputs (1 byte len)
};

byte [] ra =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
  0x07, // "Write EEPROM" cmd (1 byte len)
  0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
  (byte)0x01, // All
channels Digital (1 byte len)
      // All
channels as Outputs (1 byte len)
};

```

```

// Set all digital outputs to HI level
// Write RAM variable ICP command

```

```

// Operation code = 0x06, address in RAM 0x117, 1111 1111 - all channels "1"
output
// Each bit of the last byte represents each channel state accordinly
byte [] sa0 =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
  0x06, // "Write EEPROM" cmd (1 byte len)
  0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
  (byte)0x00, // All
channels lo except 5,6,7 (1 byte len)
};
byte [] sa1 =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
  0x06, // "Write EEPROM" cmd (1 byte len)
  0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
  (byte)0x8f, // All
channels HI (1 byte len)
};
byte [] sOn =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
  0x06, // "Write EEPROM" cmd (1 byte len)
  0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
  (byte)0x7f, // All
channels HI (1 byte len)
};
byte [] sOff =
{ 0x00, 0x00, 0x00, 0x00, 0x00, // Password (5 bytes len)
  0x06, // "Write EEPROM" cmd (1 byte len)
  0x00, 0x00, 0x01, 0x17, // Start location (4 bytes len)
  (byte)0x0f, // All
channels HI (1 byte len)
};
byte [] resp = new byte[1024]; // Network response buffer

```

```

public Start(){

    setLayout(new BorderLayout());
    Panel p = new Panel();
    Panel p2 = new Panel();
    p2.setLayout(new FlowLayout());
    p.setLayout(new BorderLayout());
    read.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            Start.this.readK();
        }
    });
    select.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            Start.this.selectK();
        }
    });
    on.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            Start.this.tOn();
        }
    });
    off.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){

```



```

        Start.this.tOff();
    }
});

value.setSize(new Dimension(0,200));
p.add("North",read);
p.add("South",select);
p2.add("North",on);
p2.add("South",off);
p.add("Center",value);
add("North",p);
add("South",p2);
add("Center",eCond);

}

private void tOn(){

    try{
        sOut.write(sOn);
        sIn.read (resp);

    }
    catch (Exception e)
    {
        e.printStackTrace();
        addItem(" Start ");
    }

}

private void tOff(){

```

```
try{
sOut.write(sOff);
sIn.read (resp);

}
catch (Exception e)
{
    e.printStackTrace();
    addItem(" Start ");
}
}
```

```
private void selectK(){
```

```
int i,j;
try{
sOut.write(sa1);
sIn.read (resp);

for(i=0;i<100000;i++)
    ;
for( j=0;j<100000;j++)
    ;

sOut.write(sa0);
sIn.read (resp);

for(i=0;i<100000;i++)
    ;
for( j=0;j<100000;j++)
    ;
}
```

```

        catch (Exception e)
        {
            e.printStackTrace();
            addItem(" Start ");
        }
    }

private void readK(){

    int i,j;
    int v=0;
    int ec;
    try{

        // Send ICP command - Configure all channels as digital outputs
        sOut.write(setAllDigitalOutput);

        // Read and ignore replay byte from the Ipsil device
        sIn.read(resp);

        sOut.write(ra);

        sIn.read(resp);
        ec= resp[8]&0x10;
        this.sa0[10]=(byte)((this.sa0[10]&0xef)|resp[8]);
        this.sa1[10]=(byte)((this.sa1[10]&0xef)|resp[8]);
        if(ec==16)
            eCond.setText("OFF");
        else if(ec==0)
            eCond.setText("ON");
        v+=(resp[8]&0x0f)*1000;
        addItem(" FirstDigit: "+(resp[8]-32));
    }
}

```

```

// Send ICP command - Set HI all on all of the channels
sOut.write(sa1);
// Read and ignore replay byte from the Ipsil device
sIn.read (resp);
for(i=0;i<100000;i++)
;
for( j=0;j<100000;j++)
;
sOut.write(sa0);
// Read and ignore replay byte from the Ipsil device
sIn.read (resp);

sOut.write(ra);

sIn.read(resp);

v+=(resp[8]&0x0f)*100;
addItem(" SecondDigit: "+(resp[8]-32));

// Send ICP command - Set HI all on all of the channels
sOut.write(sa1);
// Read and ignore replay byte from the Ipsil device
sIn.read (resp);

for(i=0;i<100000;i++)
;
for( j=0;j<100000;j++)
;

sOut.write(sa0);
// Read and ignore replay byte from the Ipsil device
sIn.read (resp);

```

```

sOut.write(ra);

sIn.read(resp);

v+=(resp[8]&0x0f)*10;
addItem(" ThirdDigit: "+(resp[8]-32));

// Send ICP command - Set HI all on all of the channels
sOut.write(sa1);
// Read and ignore replay byte from the Ipsil device
sIn.read (resp);

for( i=0;i<100000;i++)
;
for( j=0;j<100000;j++)
;

sOut.write(sa0);
// Read and ignore replay byte from the Ipsil device
sIn.read (resp);

sOut.write(ra);

sIn.read(resp);

v+=resp[8]&0x0f;
addItem(" ForthDigit: "+(resp[8]-32));

for(int k=0;k<7;k++){
sOut.write(sa1);
sIn.read (resp);

```

```

        for(i=0;i<100000;i++)
            ;
        for( j=0;j<100000;j++)
            ;

        sOut.write(sa0);
        sIn.read (resp);

        for(i=0;i<100000;i++)
            ;
        for( j=0;j<100000;j++)
            ;
    }
}
catch (Exception e)
{
    e.printStackTrace();
    addItem(" Start ");
}

value.setText((v*2)+"KWH");
}

public void init()
{
    try
    {
        //ia = InetAddress.getByName("192.168.1.101");
        //ia = InetAddress.getLocalHost();
        //addItem(ia.getHostAddress());
    }
}

```

```

        ia = InetAddress.getByName( getCodeBase().getHost() );

        // Init TCP socket. Connect to the device itself on Ipsil port
        sock = new Socket(ia, IPSIL_PORT);
        addItem(ia.toString());

        // Create input data stream
        sIn = sock.getInputStream();

        // Create output data stream
        sOut = sock.getOutputStream();

    }

    catch (Exception e)
    {
        e.printStackTrace();
        addItem(" Init " );
    }

}

private void addItem(String newWord)
{
    System.out.println(newWord);
    buffer.append(newWord);
    repaint();
}

public static void main(String args[]) {

```

References

```
Frame f = new Frame("KWH Reading");
Start s = new Start();
s.init();
s.start();

f.add("Center", s);
f.setSize(800, 400);
f.show();
}

public void start()
{

}

} //end of the applet
```


References

1. PPU Projects

- [1] "Micro Web-Server"
- [2] "Micro-Web Server for controlling and monitoring applications"
- [3] "Controlling a Digital Camera Using Micro Web Server"

2. International Projects

- [1] "Micro Web server for domestic application"
- [2] "Electric Meter Reading via Bluetooth"
- [3] "Power Meter Reading through internet connection"

3. Books

- [1] "Data Communication and Networking, 2nd edition update, Behrouz A . Forouzan, McGraw-Hill, ISBN: 0-07-282294-5"

4. Other Sources

- [1] IPU8930 Developer Guide
- [2] Electronic instrumentation and measurement techniques, 2nd edition by William David Cooper
- [3] The IPSIL web site,"<http://www.ipsil.com>"
- [4] Electricity Meter Reading via Bluetooth, a project at the Nanyang Technological University.
- [5] Cirrus Logic web site,"<http://www.cirrus.com>"
- [6] Archnet web site,
"http://www.archnetco.com/english/product/product_s1.htm"
- [7] Power measurement site,"<http://www.pwr.com>"